1990-09

The use of neural networks in adaptive control

Nedresky, Donald L.

Monterey, California: Naval Postgraduate School

(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THE USE OF NEURAL NETWORKS IN
ADAPTIVE CONTROL

by

Donald L. Nedresky

SEPTEMBER, 1990

Thesis Advisor: Prof D.J. Collins

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate Schoo 6b OFFICE SYMBOL (If applicable) 31	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)

The Use of Neural Networks in Adaptive Control

12 PERSONAL AUTHOR(S)
Nedresky, Donald L.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
	FROM _____ TO _____	1990 September	67

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Neural Networks, Adaptive Control, Backpropagation, Parameter Estimation, Parallel Distributed Processing

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

An adaptive control system is a system that detects changes in the plant parameters and makes necessary adjustments to the systems performance. This thesis examines the use of parallel distributed processing systems (neural networks) in adaptive control. A general neural network structure is introduced and a description of the Backpropagation paradigm is given. A discussion of adaptive control theory including the one step ahead prediction control algorithm and the linear least squares estimation is given. A neural network structure consistent with adaptive control theory is developed and tested by simulating the lateral and directional motion of the A-4 aircraft. The network output is then compared to the output of the true system. The purpose of this thesis is to develop and test a neural network structure capable of performing the parameter estimation and control functions of an adaptive controller.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Dr. D. J. Collins	22b TELEPHONE (Include Area Code) (408)646-2826 22c OFFICE SYMBOL 67Co

DD Form 1473, JUN 86 *Previous editions are obsolete* SECURITY CLASSIFICATION OF THIS PAGE

S/N 0102-LF-014-6603 UNCLASSIFIED

Approved for public release; distribution is unlimited.

THE USE OF NEURAL NETWORKS IN ADAPTIVE CONTROL

by

Donald L. Nedresky

B.S., Indiana University of Pennsylvania
M.S., University of Southern California

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September, 1990

Author: [redacted]
Donald L. Nedresky

Approved by: [redacted]
Daniel J. Collins, Thesis Advisor

[redacted]
Louis V. Schmidt, Second Reader

[redacted]
E. Roberts Wood, Chairman
Department of Aeronautics and Astronautics

ABSTRACT

An adaptive control system is a system that detects changes in the plant parameters and makes necessary adjustments to the systems performance. This thesis examines the use of parallel distributed processing systems (neural networks) in adaptive control. A general neural network structure is introduced and a description of the Backpropagation paradigm is given. A discussion of adaptive control theory including the one step ahead prediction control algorithm and the linear least squares estimation is given. A neural network structure consistent with adaptive control theory is developed and tested by simulating the lateral and directional motion of the A-4 aircraft. The network output is then compared to the output of the true system. The purpose of this thesis is to develop and test a neural network structure capable of performing the parameter estimation and control functions of an adaptive controller.

TABLE OF CONTENTS

I. INTRODUCTION

An effective control system must be able to adapt to factors that effect the dynamics of the system such as changes in the operating environment or component wear with time. The control system having a candid ability of adaptation (that is, the control system itself detects changes in the plant parameters and makes necessary adjustments to the performance) is called the adaptive control system [Ref. 1:pp. 5]. Previous research described in [Ref. 2] demonstrated how adaptive control problems can be represented using neural networks. The system used to investigate the use of neural networks for estimation and control was the longitudinal motion of the A-4 aircraft.

This thesis will study the use of neural networks for estimation and control using the lateral/directional equations of motion of the A-4 aircraft. Chapter II will discuss what neural computing is and describe neural network processing. A description of the back-propagation algorithm will also be given. Chapter III will discuss adaptive control theory and the relationship between neural networks and adaptive control. Chapter IV will discuss the experimental setup including the hardware and software used, a description of the lateral/directional motion of the A-4 aircraft and development of the neural network model used to simulate that motion. Chapter V includes the results and discussion of the simulation experiment.

II. NEURAL NETWORK THEORY

A. WHAT IS NEURAL COMPUTING?

Many models have been developed that try to replicate the information processing tasks of the human brain. The study of neural computing involves the use of computer models to perform this replication in a very simplified manner. These models, which are a form of parallel distributed processes (PDP), are known as neural networks.

B. ANALOGY TO THE BRAIN

Although their methods of operation may differ significantly from that of the brain, all neural network models can be described through an analogy to the components of the brain which they attempt to model. Figure 1 is a model of the basic building block of the human nervous system, (the neuron). Its nucleus is a processing unit which receives impulses from other neurons through input paths called dendrites. If the input signal is strong enough, the neuron is activated, and an output signal is generated. This output signal is transmitted through paths called axons. The axon splits into multiple paths which connect to the dendrites of other neurons through junctions called synapses. These junctions are chemical in nature. The magnitude of the signal transferred depends on the amount of chemicals released by the axons and received by the dendrites. This synaptic efficiency or strength is what is modified when the brain learns. The synapse combined

with the processing of information in the neuron form the basic memory mechanism of the brain. [Ref. 3:pp. 3-4]

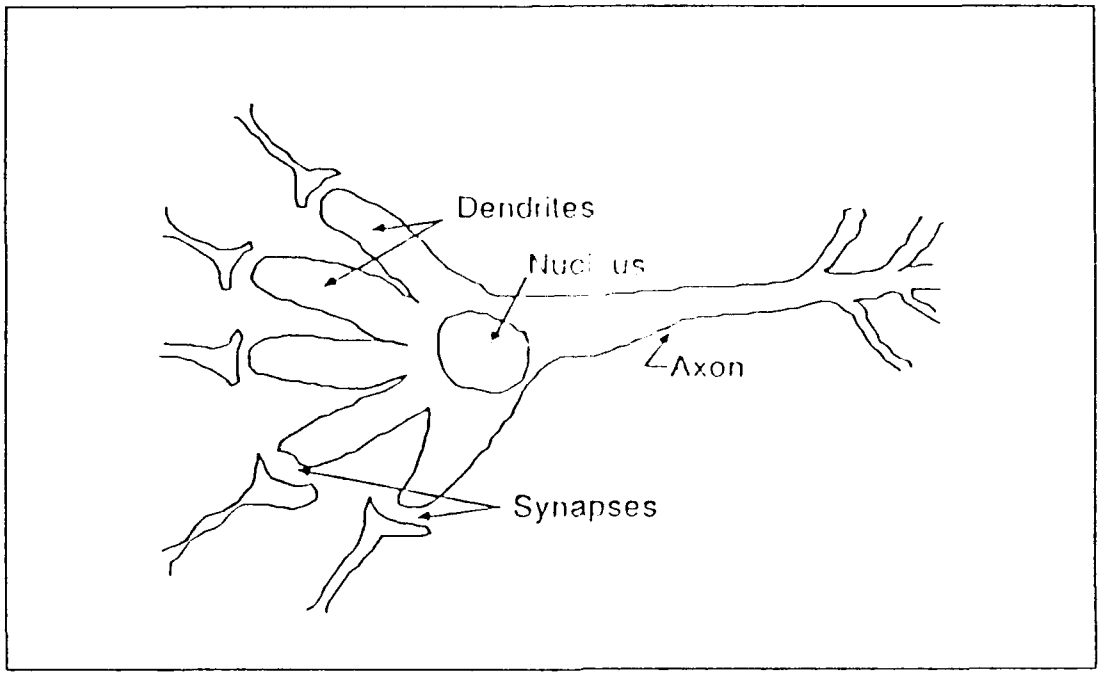

Figure 1: Basic Building Block of the Nervous System [Ref. 3]

C. WHAT IS A NEURAL NETWORK?

The basic components of a parallel distributed processing system (ie., a neural network), are shown in Figure 2. Each circle represents a processing unit with an activation value a(t). This activation is processed by an activation function which produces an output o(t). Each output passes along a connection which passes on that output to other processing elements much the same way as the axons do to dendrites in the human brain. Each connection is weighted to determine the effect the output will have on the connecting processing element. All of the inputs to a processing element are

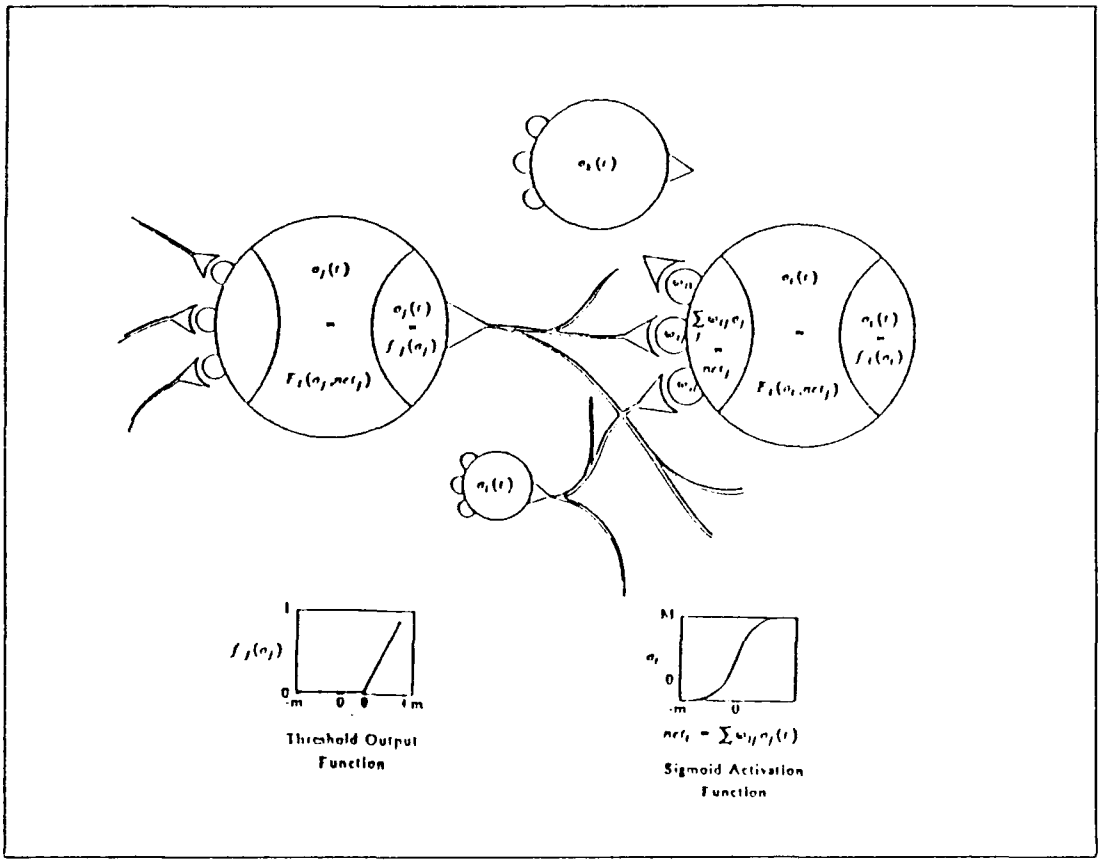

Figure 2: Basic Components of a PDP System [Ref. 3]

combined with the current activation value through an activation function F which determines the new activation level for that particular processing element.

D. HOW DOES A NEURAL NETWORK WORK?

Figure 3 is a simplified representation of a neural network architecture. The network consists of a number of processing elements all connected and weighted as described above. The elements of the network are usually grouped into layers and are randomly or fully connected as shown. The first layer is referred to as the input buffer and is the point at which data is presented to the network. The output buffer holds the

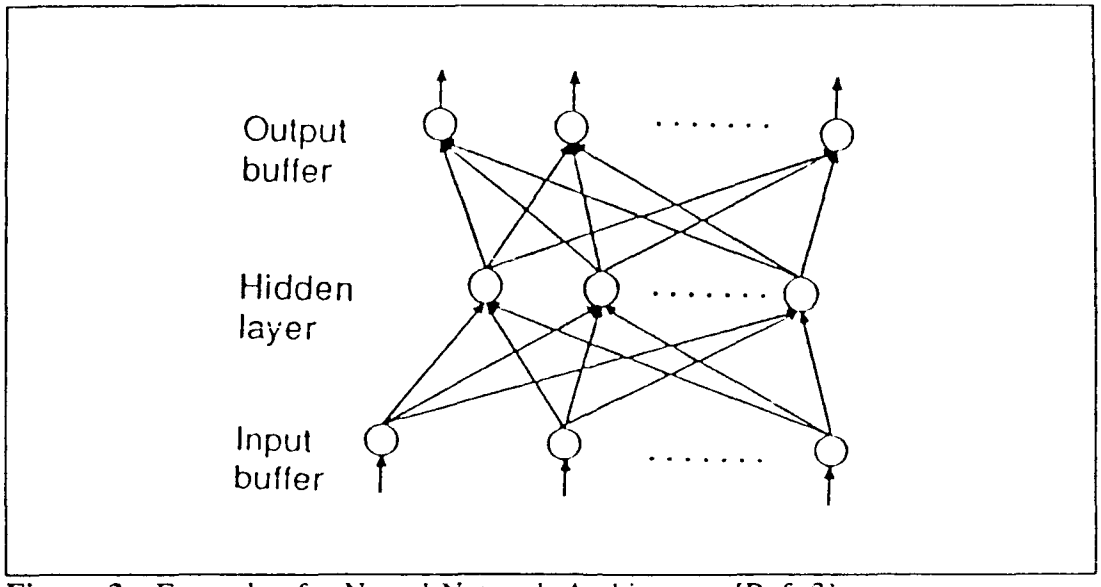

Figure 3: Example of a Neural Network Architecture [Ref. 3]

data produced as a result of processing a given input. Additional layers between the input and output layers are referred to as hidden layers.

As mentioned earlier, learning in the human brain takes place when the strength of the output signal transferred from one processing element across the synapse to another processing element is increased. This is similar to the learning process occurring in a neural network. As input data is presented to the network through the input buffer it is propagated to the output buffer, and the output obtained is compared to the desired output. The error between the desired output and the actual output is calculated and the connection weights are altered based upon this error until the desired output is obtained. In this way learning takes place in the neural network in much the same way it does in the human brain.

Learning can be classified as supervised or unsupervised. In supervised learning the desired response to a given input is presented to the output buffer. If the desired

output is the same as the input, the network is called auto-associative. If it is different from the input it is called hetero-associative. If the desired output is not shown to the output buffer unsupervised learning takes place.

E. THE BACK-PROPAGATION ALGORITHM

A typical network based upon the back-propagation algorithm has a structure similar to that shown in Figure 3. It consists of an input layer, an output layer and at least one hidden layer. A typical processing element in a back-propagation network is shown in Figure 4. The notation in brackets symbolizes what network layer is being considered. The notation $x_j^{[s]}$ represents the existing output level of the j^{th} element in layer s. $I_j^{[s]}$

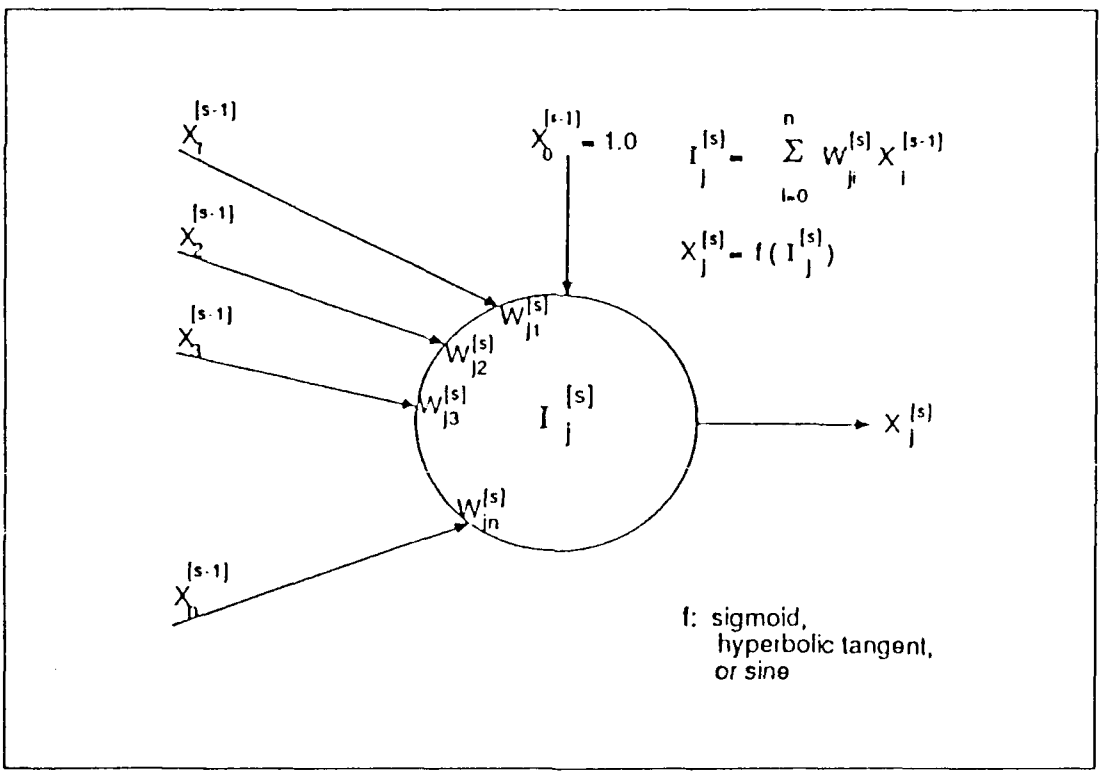

Figure 4: A Typical Back-Propagation Processing Element [Ref.3]

represents the weighted sum of all inputs to the j^{th} element in layer s and $w_{ji}^{[s]}$ is the value of the connection weight between the i^{th} element in layer (s-1) to the j^{th} element in layer s.

The back-propagation processing element transfers its inputs to other processing elements using the following relationship:

$$I_j^{[s]} = \sum_{i=0}^{n} w_{ij}^{[s]} x_i^{[s-1]} \qquad (2.1)$$

$$x_j^{[s]} = f(\sum_i (w_{ji}^{[s]} \times x_i^{[s-1]})) = f(I_j^{[s]})$$

The symbol f represents the transfer function, usually the sigmoid function, but can be any differentiable function. The sigmoid function is represented in z transform notation by:

$$f(z) = (1.0+e^{-z})^{-1} \qquad (2.2)$$

1. Global Error Minimization

A network using the back-propagation algorithm can be assumed to have a global error function which is a differentiable function of all connection weights in the network. The error that is passed back through the layers is:

$$e_j^{[s]} = -\partial E / \partial I_j^{[s]} \qquad (2.3)$$

which is a measure of the local error for each element in layer s. In order for learning to take place, the global error E must be minimized by modifying the weights in the back-

propagation network. If a vector i which produces an output o is input to the network, and a desired output d is specified, then the global error in achieving the desired output is

$$E = 0.5 * \sum_k ((d_k - o_k)^2) \tag{2.4}$$

where k is used to index the components of d and o and the raw local error is d_k-o_k. The scaled local error is given by

$$\begin{aligned} e_k^{(0)} &= -\partial E / \partial I_k^{(o)} \\ &= -\partial e / \partial o_k * \partial o_k / \partial I_k \\ &= (d_k - o_k) * f'(I_k) \end{aligned} \tag{2.5}$$

To modify the network weights the following gradient descent rule is used:

$$\Delta w_{ji}^{[s]} = -lcoef * (\partial E / \partial w_{ji}^{[s]}) \tag{2.6}$$

This equation states that each weight is changed by multiplying the magnitude and direction of the negative gradient on the error surface by the learning coefficient lcoef. From the chain rule and equation (2.1) the change in the connection weights can be determined by

$$\Delta w_{ji}^{[s]} = lcoef * e_j^{[s]} * x_i^{[s-1]} \tag{2.7}$$

The above relationship only applies to the output layer. The error signal for hidden units which do not have a target output is computed recursively in terms of the error signals of the units to which it directly connects and the weights of those connections. [Ref. 4:pp. 327] This relationship is given mathematically as:

$$e_k^{[s]} = f'(I_k)\Sigma_k w_k * e_k \tag{2.8}$$

An in-depth discussion of the back-propagation algorithm is given in [Ref. 3:pp. NC111-NC131].

2. Back-Propagation Algorithm Summary

If i denotes the input vector and the desired output is d, a general summary of the steps in the standard back-propagation algorithm is given as follows:

- Propagate i from the input layer through the network to the output layer to obtain an output o. The summed inputs I_j and output states x_j for each processing element are set as the information is propagated through the network.
- Calculate the scaled local error using (2.5) and the delta weight using (2.7) for each processing element in the output layer.
- Calculate the scaled local error and the delta weight using (2.7) for each layer s starting at the layer preceding the output layer and ending with the layer above the input layer.
- Modify all network weights by adding the delta weights to the previous weights. [Ref. 3:pp. NC-116]

3. Fast Back-Propagation

This thesis uses the fast back-propagation control strategy available in NEURALWORKS PROFESSIONAL II. The fast back-propagation algorithm is a variation to the back-propagation algorithm that was presented by Tariq Samad at the 1988 INNS Conference. The fast back-propagation algorithm replaces equation (2.7) with:

$$\Delta W_{ji}^{[s]} = lcoef * e_j^{[s]} * (x_i^{[s-1]} + e_i^{[s-1]}) \tag{2.9}$$

which states that the error at layer [s-1] and the activation value are added before updating the weight. Equation (2.9) can be expanded to show the second order relationship such that:

$$\Delta w_{ji} = lcoef * e_j^{[s]} * x_i^{[s-1]} + lcoef * e_j^{[s]} * e_i^{[s-1]} \tag{2.10}$$

Another form of equation (2.10) proposed by Samad adds a multiple, k, of the error to the activation value:

$$\Delta W_{ji}^{[s]} = lcoef * e_j^{[s]} * (x_i^{[s-1]} + k * e_i^{[s-1]}) \tag{2.11}$$

Using fast back-propagation can drastically reduce the number of iterations required to reach convergence. [Ref. 3:pp. NC120-NC121]

III. ADAPTIVE CONTROL THEORY

The dynamic characteristics of most control systems are seldom constant. Typical feedback control systems are capable of attenuating the effects of small changes in those dynamic characteristics. However, in order to deal with large changes in the system parameters and environment, the control system must have the ability of adaptation [Ref. 1:pp. 5]. The design of an adaptive control system is conceptually simple. It includes combining a particular parameter estimation technique with any control law. This approach of using the estimates as if they were the true parameters for the purpose of design is called certainty equivalence adaptive control [Ref. 4:pp. 180]. Figure 5 contains a block diagram of a general adaptive control system. A great many different algorithims can be created depending upon the parameter estimation technique and control law being used. In this thesis only the control and estimation of deterministic systems will be examined. A deterministic system is one in which the system response is completely described by the model and which modelling errors are not significantly affected by noise. The one step ahead control algorithm will be used to develop the control model while the back-propagation algorithm will be used as a basis for developing the estimation portion of the adaptive controller. A more detailed explanation of adaptive control and the one step ahead algorithm is given in [Ref. 4].

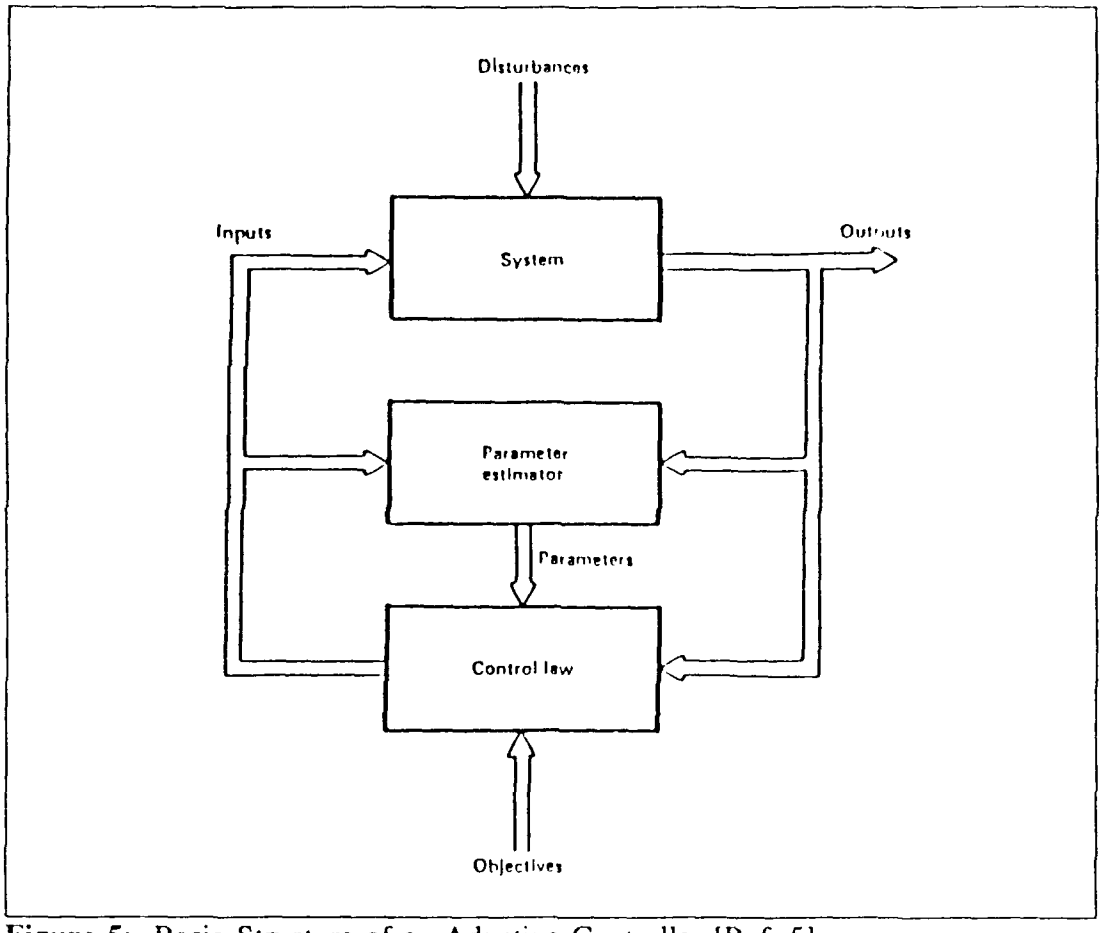

Figure 5: Basic Structure of an Adaptive Controller [Ref. 5]

A. ONE STEP AHEAD PREDICTION CONTROL

One step ahead prediction control brings the output y(t+d), where d represents a time delay, to a desired output value $y^*(t+d)$ in one step. For most control systems, it is assumed that the system is linear and finite dimensional. As described in [Ref. 4:pp. 32-33], the simplest model used to develop adaptive control algorithms for this type of system is the deterministic autoregressive moving-average (DARMA) model. The DARMA model can be described by the formula

$$
\begin{aligned}
A(q)y(t) &= B(q)u(t) \\
&\textit{where:} \\
A(q^{-1}) &= I+A_1(q)+\ldots+A_n(q) \\
B(q^{-1}) &= B_o+\ldots+B_m(q)
\end{aligned}
\tag{3.1}
$$

and A(q) and B(q) represent matrix polynomials written in backward shift operator notation, q^{-1}. The system output and input are represented by the discretized form of y(t) and u(t) respectively. The DARMA model is equivelent to an observable state-space model and can describe the input-output properties of a general state space model having an arbitrary initial state [Ref. 4:pp. 32]. Expanding the single input single output (SISO) DARMA model in the shift operator and rearranging equation (3.1) gives

$$
y(t) = b_1u(t-1)+b_2u(t-2)..-a_1y(t-1)-a_2y(t-2).. \tag{3.2}
$$

which can be used as a predictor for the output at the next time step

$$
y^*(t+1) = b_1u(t)+b_2u(t-1)...-a_1y(t)-a_2y(t-1)... \tag{3.3}
$$

where $y^*(t+1)$ is the predicted value of y(t+1). By replacing $y^*(t+1)$ with the desired output $y_d(t+1)$ and solving for u(t), the control required to bring the system to a desired value in one step is given as:

$$
u(t)=\frac{1}{b_1}[y_d(t+1)+a_1y(t)+a_2y(t-1)..-b_2u(t-1)..] \tag{3.4}
$$

This equation represents the one step ahead control law. [Ref. 2:pp. 17-20] The term $y_d(t+1)$ in equation 3.4 can represent a reference input to the system. If the past input and output values, u(t) and y(t) are state variables, the one step ahead controller can be thought of as a state variable feedback with a reference input controller [Ref. 2:pp. 19].

$$u(t) = K(t)\underline{x}(t)+r(t) \tag{3.5}$$

The state variable in equation (3.5) is made up of the vector containing the inputs and outputs of equation (3.4) and provides a controller for an adaptive algorithm [Ref. 4:pp. 118-171]. Using the vector of past input and output measurements as some state vector, the idea of a controller based on the weighted sum of state variables and a reference input may be developed [Ref. 2:pp. 20]. The input to a neural network processing element is defined as the weighted sum of all inputs from processing elements connected to it. Therefore a processing element in a neural network is capable of representing this type of controller.

B. LINEAR LEAST SQUARES ESTIMATION

As previously mentioned, adaptive control consists of two functions, estimation and control. A derivation of one form of the linear least squares estimator, the recursive least squares method, was performed to develop a general structure for estimation in [Ref. 2]. The general form of the predictor corrector equation for all least squares parameter estimation schemes was given as:

$$\theta(t+1) = \Phi(t)+M(t)\Phi(t-1)e(t) \tag{3.6}$$

where:
M(t)..is the algorithm gain
Φ(t-1)..is the regression vector
e(t)..is the model prediction error

The back-propagation learning rule discussed earlier is similar to the general form of the linear least squares parameter estimator in equation (3.6). In fact, a neural network with

a linear activation function is a parallel distributed processing implementation of the general linear least squares estimator [Ref. 2:pp. 22]. Therefore, the theorems and proofs that are applicable to least squares estimation are in a general sense applicable to a network using the back-propagation algorithm.

By combining the control and estimation algorithms described in this and the previous section, an adaptive control structure was developed in [Ref. 2]. Because of its similarities with this structure, the back-propagation neural network was used to produce models for neural network adaptive control. Neural network adaptive control structures were developed for the longitudinal motion of the A-4 aircraft. In this thesis, those models will be applied to the lateral/directional equations of motion of the A-4 aircraft.

IV. EXPERIMENTAL SETUP

A. HARDWARE AND SOFTWARE USED

All data processing for this thesis was done on the Sun 386i/250 workstation using the Neuralworks Professional II software package by Neuralware, Inc. The Sun 386i/250 workstation uses a full 32-bit architecture with a 25 MHz Intel 80386 central processing unit (CPU). It has an XP cache memory card with 4 MBytes of main memory for performance that exceeds five million instructions per second (MIPS). The configuration used for this thesis included 16 MB of memory, a VGA adapter, a 16 inch high resolution color monitor, one 3.5 inch floppy disk drive and a 0.25 inch tape drive. The Sun operating system provides a windowed environment which allows multiple tasks to be performed at the same time, greatly enhancing the flexibility of the system.

The software used for this thesis, Neuralworks Professional II, offers the capability of designing over a dozen different types of networks. Neuralworks supports a general file format for reading data into the network that encompasses standard spreadsheet file formats. Input-output may also be accomplished through keyboard interface, formatted ASCII files or user defined modules [Ref. 3:pp. UG215-UG250]. Different variables such as weights, error values, and activation levels can be displayed graphically through the use of probes which monitor and display the activation values of these parameters.

As mentioned earlier, Neuralworks provides the user with the capability of accessing the network through the use of a user defined module. This method of access is achieved

by attaching a user written procedure to the network [Ref. 3:pp. UG237]. The source code for this procedure, the USERIO program, is provided as part of the Neuralworks software package. Control strategies for input-output operations, learning and propagation are provided for all standard network types. They can also be user defined. The control strategies and USERIO programs used in this thesis are the same as those used in [Ref. 2] with the exception of the numerical values for flight conditions and numerator and denominator coefficients in the header file Transfer.txt. Complete listings of these programs can be found in [Ref. 2]. A listing of the transfer.txt file with values used in this thesis is found in Appendix A.

The transfer function numerator and denominator coefficients used as input for the networks in this simulation were obtained through the use of MATLAB, a high-performance interactive software package for scientific and engineering computation [Ref. 5:pp. 3]. In addition, MATLAB was used to perform frequency response analysis for the various flight conditions for comparison with the responses generated by the neural network.

B. LATERAL/DIRECTIONAL MOTION OF THE A-4 AIRCRAFT

The stick fixed lateral/directional motion of an airplane disturbed from equilibrium is described in detail in [Ref. 6:pp. 152-175] and [Ref. 7:pp. 353-414]. As noted in those descriptions, the lateral/directional equations of motion can be arranged in the state variable form:

$$\dot{x}(t) = A\underline{x}(t)+B\underline{u}(t)$$
$$y(t) = C\underline{x}(t)+D\underline{u}(t)$$
$$\text{where:}$$
$$\underline{x}(t) = \begin{matrix}\beta(t)..\textit{yaw } \angle \textit{ perturbation}\\ p(t)..\textit{roll rate perturbation}\\ r(t)..\textit{yaw rate perturbation}\\ \Phi(t)..\textit{roll } \angle \textit{ perturbation}\end{matrix} \qquad (4.1)$$

and the input variable is:

$$u(t) = \begin{matrix}\partial_a...\textit{aileron deflection}\\ \partial_r...\textit{rudder deflection}\end{matrix} \qquad (4.2)$$

The output variables are scaled versions of the state variables. The A and B matrices are functions of the airplane dimensional stability derivatives, mass and inertia characteristics for a given altitude and mach number. The C matrix is a scaling matrix and the D matrix is a matrix of zeros. The lateral frequency response of the A-4 airplane (ie., response due to an aileron input only) at mach .638 and 20,000 feet is presented in Figure 6. The low frequency dutch roll mode can be seen at a natural frequency of about .3 Hertz with a time period of approximately 3 seconds. The directional frequency response of the A-4 airplane (ie., response due to a rudder input only) at mach .638 and 20,000 feet is presented in Figure 7. As was done in [Ref. 2], several different linear models were used for different flight conditions in the experiment to introduce non-linearity. The linear models used are shown in Table I.

Simulation of the lateral and directional motion of the A-4 airplane was conducted by using the USERIO subprogram simo and the header file transfer.txt. Continuous state

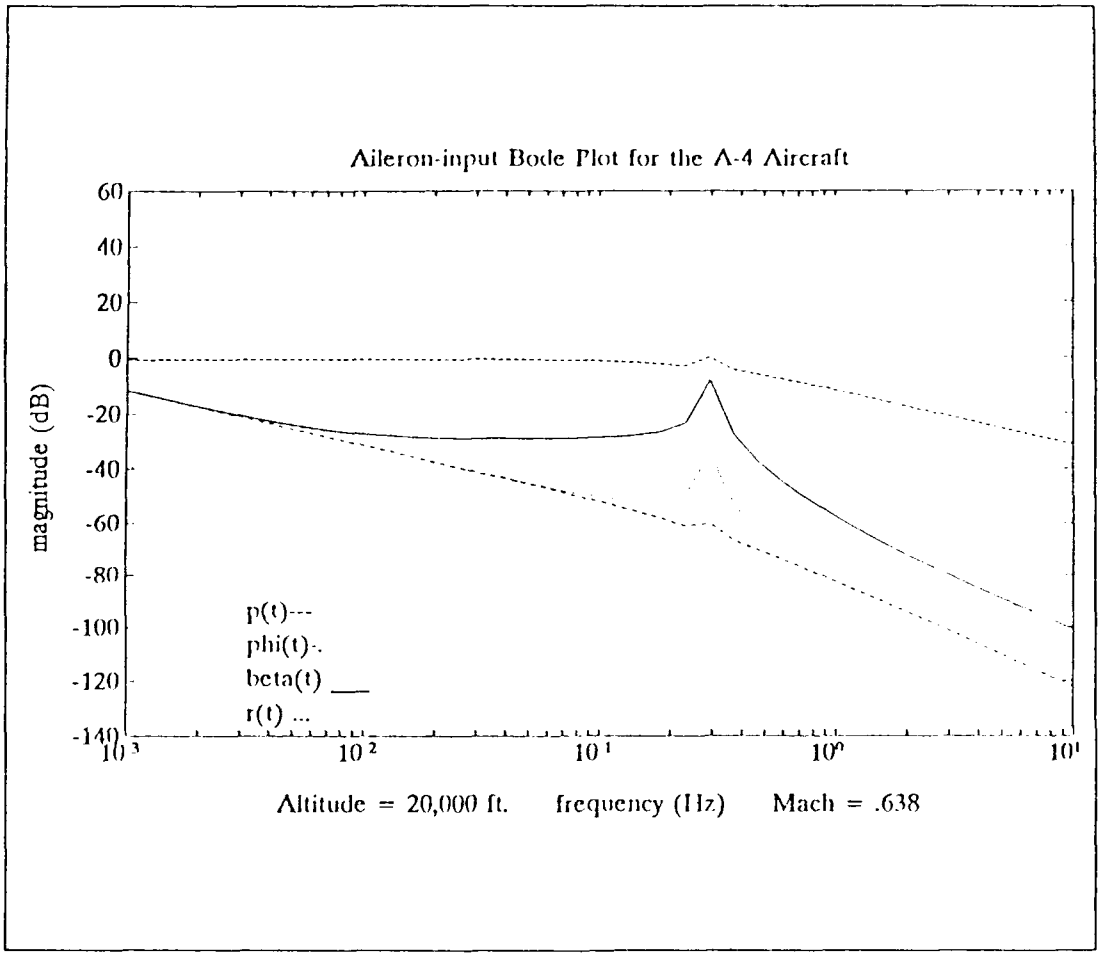

Figure 6: A-4 Lateral Frequency Response at Mach .638, 20,000 Feet

Table I: Flight Conditions Used in This Experiment.

Flight Condition	Mach #	Altitude
1	.4	sea level
2	.638	20,000 ft.
3	.5	35,000 ft.
4	.7	35,000 ft.
5	.85	sea level

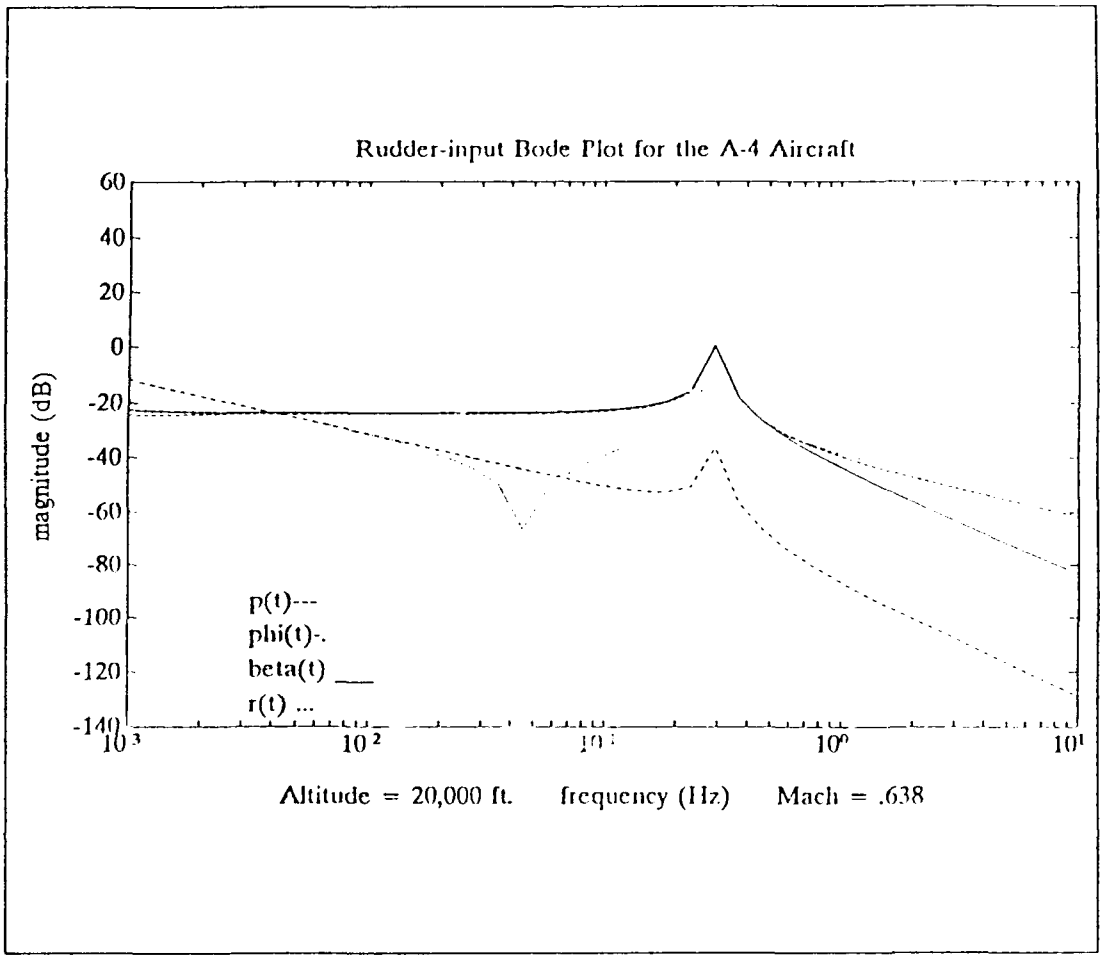

Figure 7: A-4 Longitudinal Frequency Response at Mach .638, 20,000 Feet

space models of each of the five flight conditions were produced and converted to difference equations (3.3) using the MATLAB script file shown in Appendix B. The continuous state space matrices and discrete transfer function matrices for each condition are shown in Appendix C and Appendix D. Data for the transfer.txt header file was obtained by making computations described in the MATLAB reference manual [Ref. 6:pp. C4-C26]. The matrices were converted from continuous to discrete state space form using a matrix polynomial algorithm [Ref. 2]. Next the discrete matrix was converted to a transfer function matrix using:

$$H(z)=C(zI-A)^{-1}B=Y(z)/u(z) \tag{4.3}$$

By replacing the z transform with the backward shift operator q^{-1}, the numerator and denominator terms were used to create the DARMA model:

$$A(q)y(t)=B(q)u(t) \tag{4.4}$$

Rearranging equation 4.4 gives:

$$\underline{y}(t)=B(q)u(t)-(A(q)-1)\underline{y}(t) \tag{4.5}$$

By expanding the matrix polynomials the following equations are obtained:

$$\begin{matrix} & b_{\beta 1}q^{-1}+b_{\beta 2}q^{-2}..+b_{\beta 4}q^{-4} & \\ y(t)= & b_{p1}q^{-1}+b_{p2}q^{-2}..+b_{p4}q^{-4} & u(t)-[a_1q^{-1}+a_2q^{-2}..+a_4q^{-4}]y(t) \\ & b_{r1}q^{-1}+b_{r2}q^{-1}..+b_{r4}q^{-4} & \\ & b_{\phi 1}q^{-1}+b_{\phi 2}q^{-2}..+b_{\phi 4}q^{-4} & \end{matrix} \tag{4.6}$$

By expanding the delay operator terms four recursive equations are obtained of the form:

$$y_j(t)=\sum_i [b_{ji}u(t-i)]-\sum_i [a_i y_j(t-i)] \tag{4.7}$$

where the output y_j is made up of the outputs $\beta(t)$, p(t), r(t), and $\Phi(t)$ and the a_i terms are the same for each equation. The parameters in equation (4.5) and (4.6) are used in the USERIO program to simulate the lateral motion of the A-4 aircraft. A sample of the B(q) and A((q) - 1) coefficients used for the condition of 20,000 feet and Mach .638 with aileron input only, is given in Table II.

Table II: Paramters for Flight Condition 20,000 Feet/Mach .638 With a Sampling Time of 0.1 Seconds

B(q) =	-1.2688e-04	2.6534e-04	4.3990e-05	-1.8149e-04
	1.5764e-01	-4.6480e-01	4.6208e-01	-1.5491e-01
	7.0981e-05	-2.3526e-04	2.5885e-04	-9.3603e-05
	1.4496e-05	-1.4546e-05	-1.2449e-05	1.3462e-05

A(q)-1 =	-3.7928e+00	5.4207e+00	-3.4572e+00	8.2934e-01

Note that the B(q) and A((q) - 1) terms match the numerator and denominator terms in the transfer function matrix for this flight condition shown in Appendix C. The discrete frequency response for this simulation is shown in Figure 8 while the discrete frequency response for this simulation due to a rudder input is shown in Figure 9.

C. MODEL DEVELOPMENT

A detailed description of the development of models used for this thesis is given in [Ref. 2]. The first model used was the linear neural network with no hidden layers paramaterized as four transfer functions and shown in Figure 10. The first layer, the feedback layer, consists of past values of outputs and inputs. From left to right the first three elements are past values of input $\delta(t-1)$, $\delta(t-2)$ and $\delta(t-3)$ where the delay value is noted in parentheses. The remaining elements are past output values for $\beta(t)$, $p(t)$, $r(t)$ and $\Phi(t)$. The second layer, the command layer, replicates the first with the exception

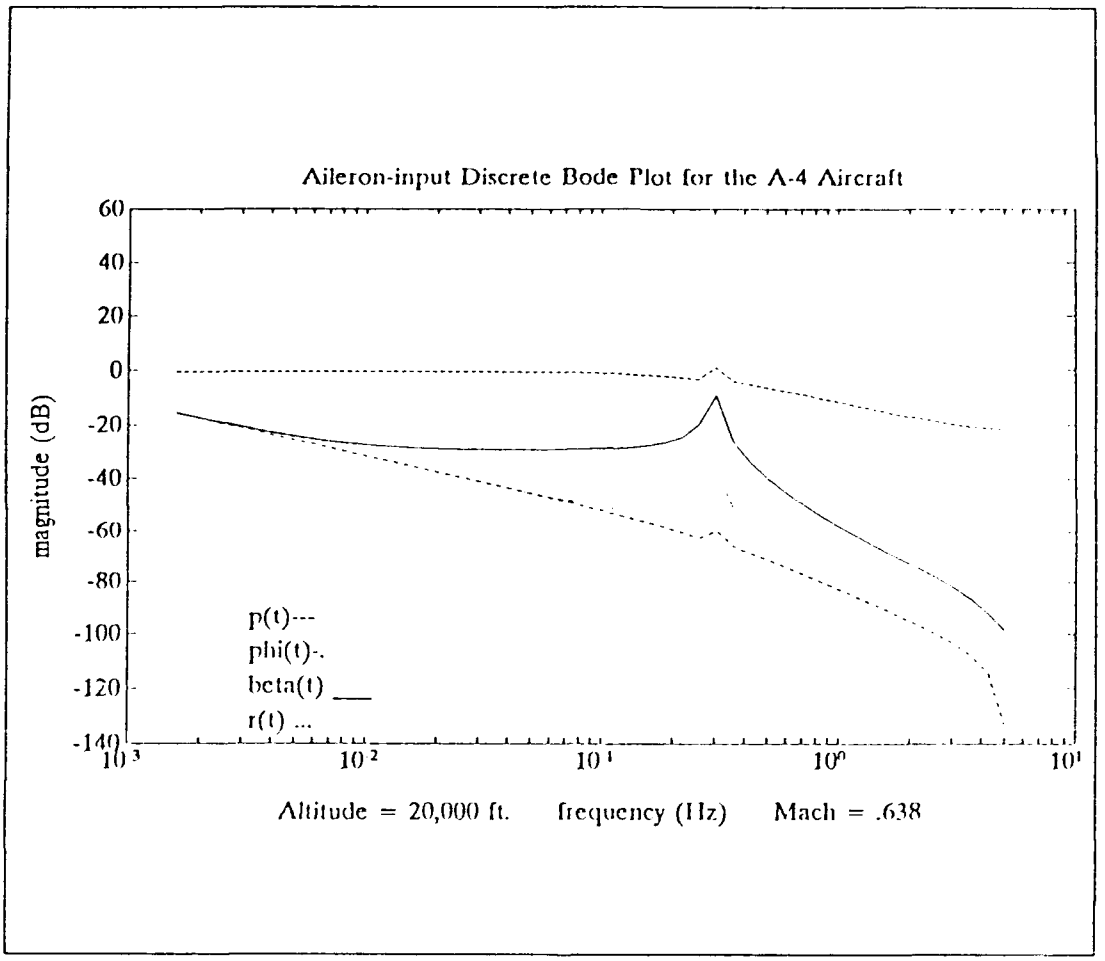

Figure 8: Discrete Frequency Response for the Lateral Motion of the A-4 Aircraft

of the reference input r(t-1), which simulates the state variable plus reference input for the control law. The third layer, the control layer, consists of one element which is a weighted sum of the reference input and the output command layer.

The output elements each have eight connections, four of which are connected to the $\delta(t)$ elements which represent the b_i terms in equations (4.5) and (4.6). Using this network structure, the weights of each output element can be compared directly to the coefficients of the true system being simulated.

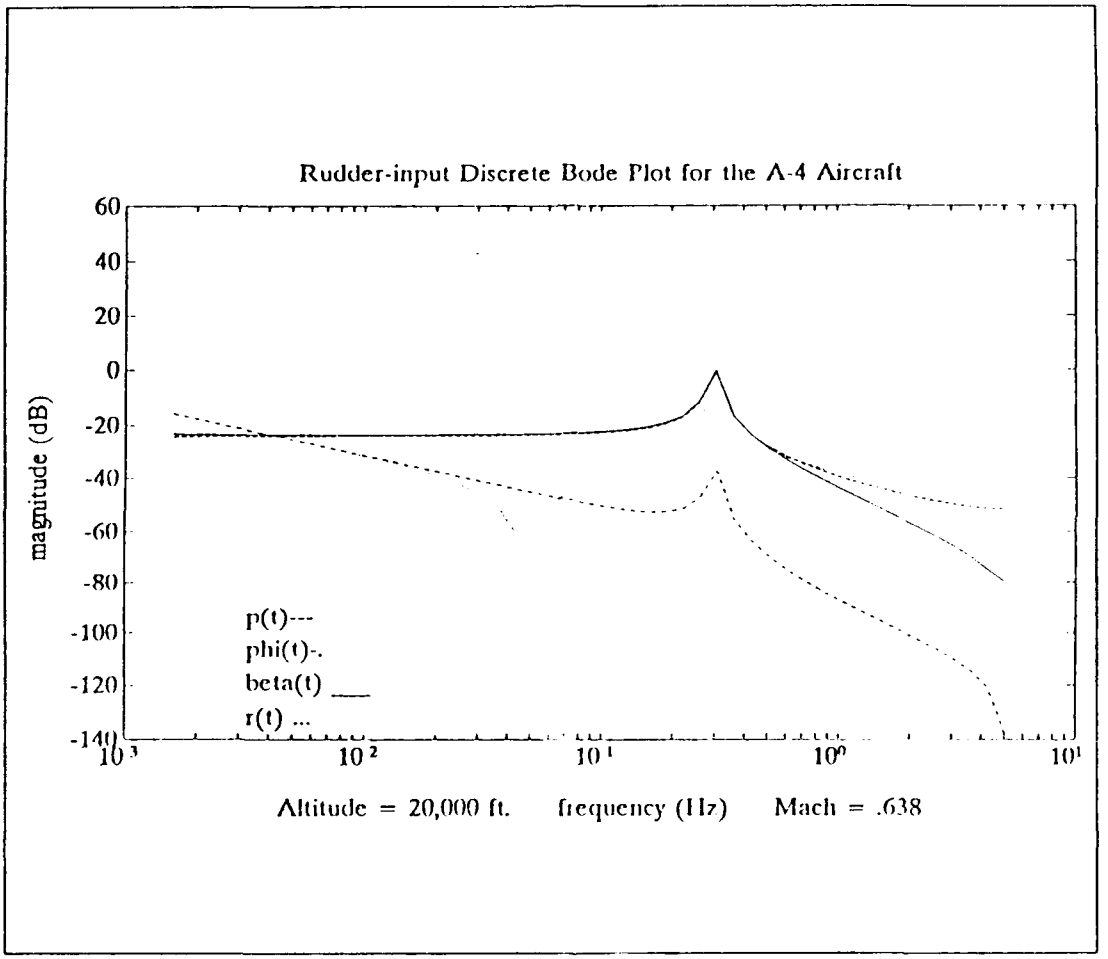

Figure 9: Discrete Frequency Response for the Longitudinal Motion of the A-4 Aircraft

For the purpose of control and estimation, this network can be considered to have no hidden layers since all inputs are directly connected to the output layer. However, when examining it from a neural networks standpoint, it can be seen that the single element in the third layer is a hidden layer through which the output of layer two is transferred to the fourth layer.

The second type of network used in this thesis is shown in Figure 11. It replicates

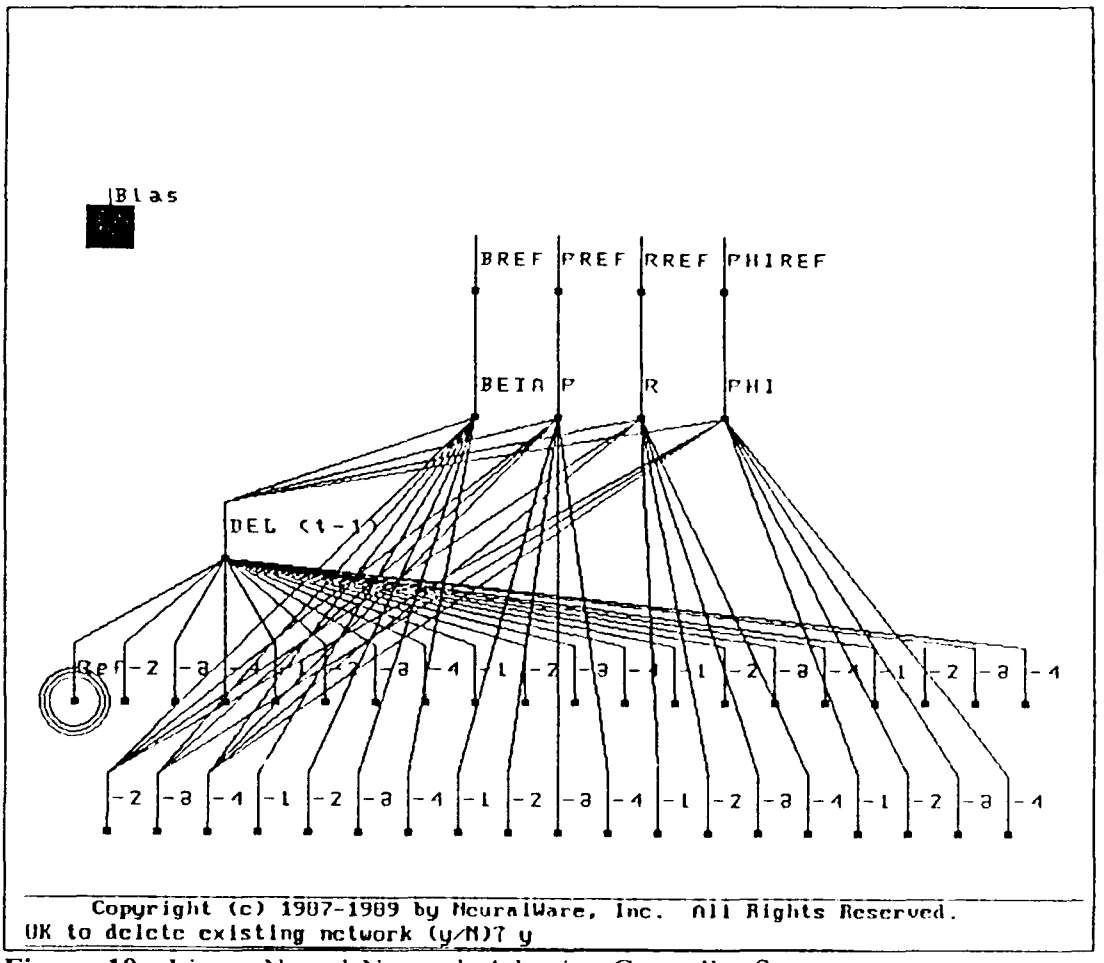

Figure 10: Linear Neural Network Adaptive Controller Structure

the linear neural network shown in Figure 10 with the exception that the output layer is fully connected to the feedback and control layers.

D. NETWORK STABILITY

Systems whose transfer functions have poles or zeros outside the unit circle in the z plane are called nonminimum phase systems. The poles and zeros of A(q) and B(q) for the simulation at condition 2 (Table I), for an aileron and rudder input are given in Table III. In the case of a simulation of an aileron input it can be seen that $\beta(t)$ and r(t) have

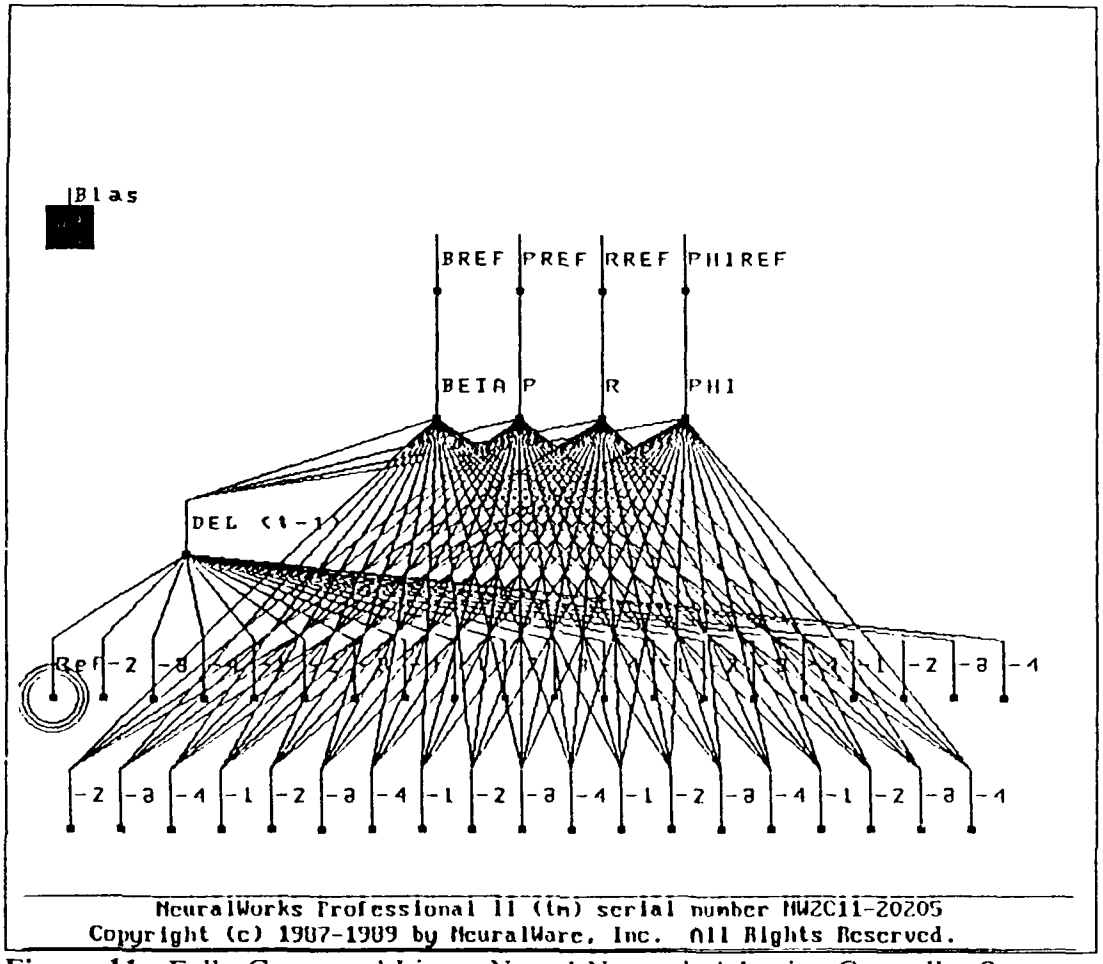

Figure 11: Fully Connected Linear Neural Network Adaptive Controller Structure

zeros outside the unit circle and p(t) has a zero on the unit circle. In the rudder input case, $\beta(t)$, p(t) and $\Phi(t)$ all have zeros outside the unit circle and r(t) has a zero very near the unit circle. These zeros are nonminimum phase. Although the simulation used in the USERIO program could generate data indefinitely, modeling errors result from the recursive nature of the simulation and the presence of nonminimum phase zeros. Errors in the simulation propagate at a rate proportional to the power of the absolute value of the system zeros. [Ref. 2:pp. 43] Since many of the zeros of the system are outside the

Table III: Poles and Zeros of the Discrete Simulation for Condition 2 with a Sampling Time of 0.1 Seconds

	aileron input	rudder input
poles =	0.9779 + 0.1859i	0.9779 + 0.1859i
	0.9779 - 0.1859i	0.9779 - 0.1859i
	1.0001	1.0001
	0.8370	0.8370
$zeros_{\beta(t)}$ =	1.8664	1.0004
	0.9951	0.8393
	-0.7702	-0.7067
$zeros_{p(t)}$ =	1.0000	1.3381
	0.9743 + 0.1830i	1.0000
	0.9743 - 0.1830i	0.7738
$zeros_{r(t)}$ =	1.2334 + 0.1858i	0.9994 + 0.0293i
	1.2334 - 0.1858i	0.9994 - 0.0293i
	0.8476	0.8372
$zeros_{\Phi(t)}$ =	0.9743 + 0.1830i	1.3347
	0.9743 - 0.1830i	0.7740
	-0.9450	-0.9300

unit circle, errors in the simulation could grow unbounded. To keep these errors from becoming signifigant, the USERIO program resets the simulation every 9000 cycles.

V. RESULTS AND DISCUSSION

This chapter will examine the results of the simulations conducted using the networks described in the previous Chapter. The ability of the linear neural network adaptive control structure in estimation will be examined. Performance of the network will be determined through the simulation of lateral and directional motion of the A-4 airplane. Finally, a fully connected networks' operation will be examined.

A. ESTIMATION DEMONSTRATION

Estimation trials were conducted by using the neural network adaptive control structure described in chapter 4 and skipping the control law synthesis phase of operation for each sample. To do this, the weights between all elements in the command layer, except the reference input, were set to zero.

1. Linear Network Estimation of Lateral Motion

The ability of the linear neural network in estimation was first examined by simulating the lateral motion of the A-4 aircraft at Mach .638/20,000 feet. The network used is shown in Figure 10 and was trained for 50,000 (5,000 seconds), 100,000 (10,000 seconds), and 200,000 cycles (20,000 seconds) using a pseudo random binary input shown in Figure 12. This input consists of a random binary input that has been bandlimited by allowing it to change every two samples versus every sample. The drop in energy above

ten Hertz limits the excitation of high frequency noise which was shown to be a problem in previous research.

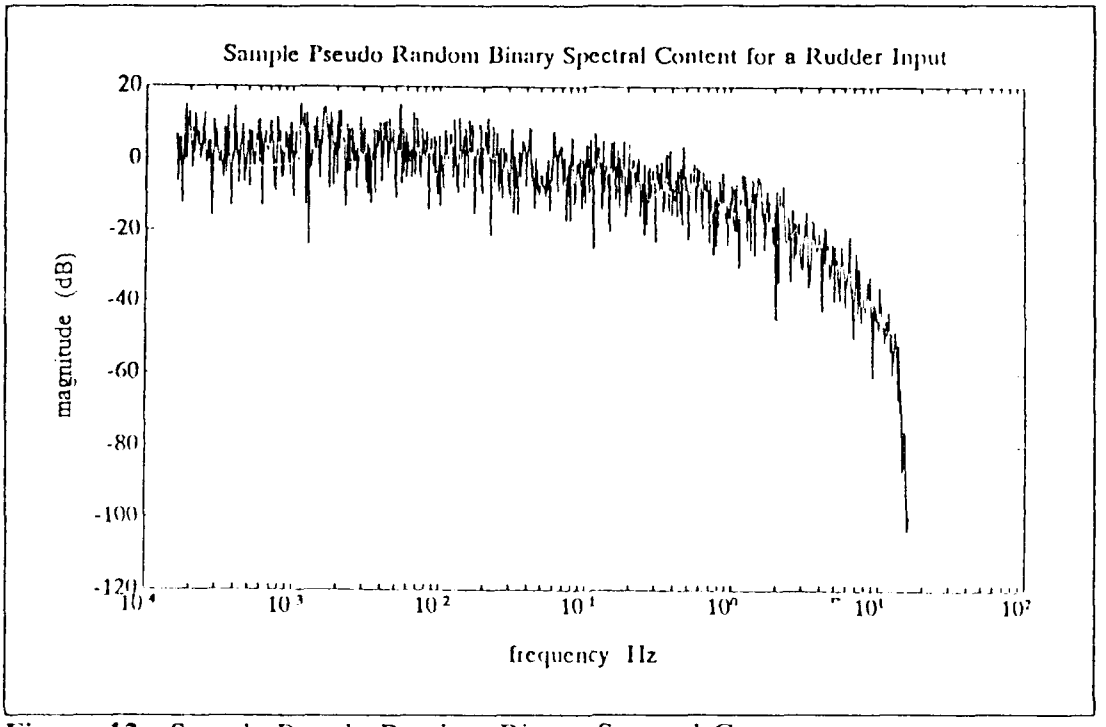

Figure 12: Sample Pseudo Random Binary Spectral Content

Because this is a linear network, the weights of the network can be equated to the numerator and denominator terms in the transfer function model of the lateral and directional equations of motion. A comparison of the weights for p(t) and $\beta(t)$, which are representative of the lateral and directional motion of the A-4 aircraft, are compared with the true system variables in Table IV.

The values of b_i and a_i determined by the network for p(t) are similar in magnitude to the true values although in some cases the signs are incorrect. This is not true for $\beta(t)$. This would seem to indicate that the network has built a better representation of those terms affected by an aileron input, (ie., lateral motion), which is

Table IV: Network Weights at 200,000 Cycles

Terms	500k Model	True Model
b_{p1}	1.556e-01	1.576e-01
b_{p2}	1.212e-01	-4.648e-01
b_{p3}	5.210e-02	4.621e-01
b_{p4}	3.440e-02	-1.549e-01
a_{p1}	-9.680e-01	-3.793e+00
a_{p2}	-2.729e-01	5.421e+00
a_{p3}	-1.115e-01	-3.457e+00
a_{p4}	-1.416e-01	8.293e-01
$b_{\beta 1}$	5.023e-02	-1.269e-04
$b_{\beta 2}$	6.290e-02	2.653e-04
$b_{\beta 3}$	-9.8[illegible]e-03	4.399e-05
$b_{\beta 4}$	1.[illegible]0e-03	-1.815e-04
$a_{\beta 1}$	6.520e-03	-3.793e+00
$a_{\beta 2}$	-1.025e-03	5.421e+00
$a_{\beta 3}$	-6.390e-02	-3.457e+00
$a_{\beta 4}$	5.830e-02	8.293e-01

being simulated in this case.

The weights of the network can be used to generate discrete-time Bode frequency response plots for comparison with the frequency response of the true system. The frequency response plots for the lateral modes of the A-4 aircraft estimation using the linear network are shown in Figure 13 through Figure 16. The response for $\beta(t)$ is shown in Figure 13. The network failed to develop a good frequency model or the proper mode shape for this parameter. However, Figure 14 shows that a much better model of the p(t) parameter was achieved. This is to be expected since the level of excitation of the p(t) and $\Phi(t)$ variables is much larger since the only input being simulated was that

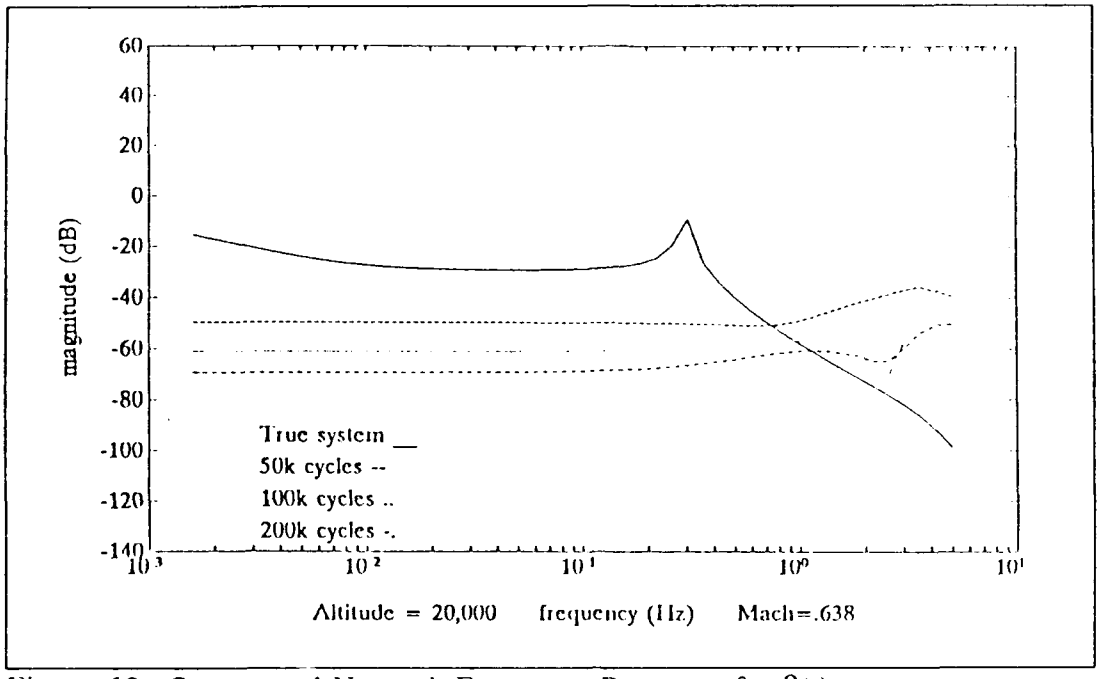

Figure 13: System and Network Frequency Response for $\beta(t)$

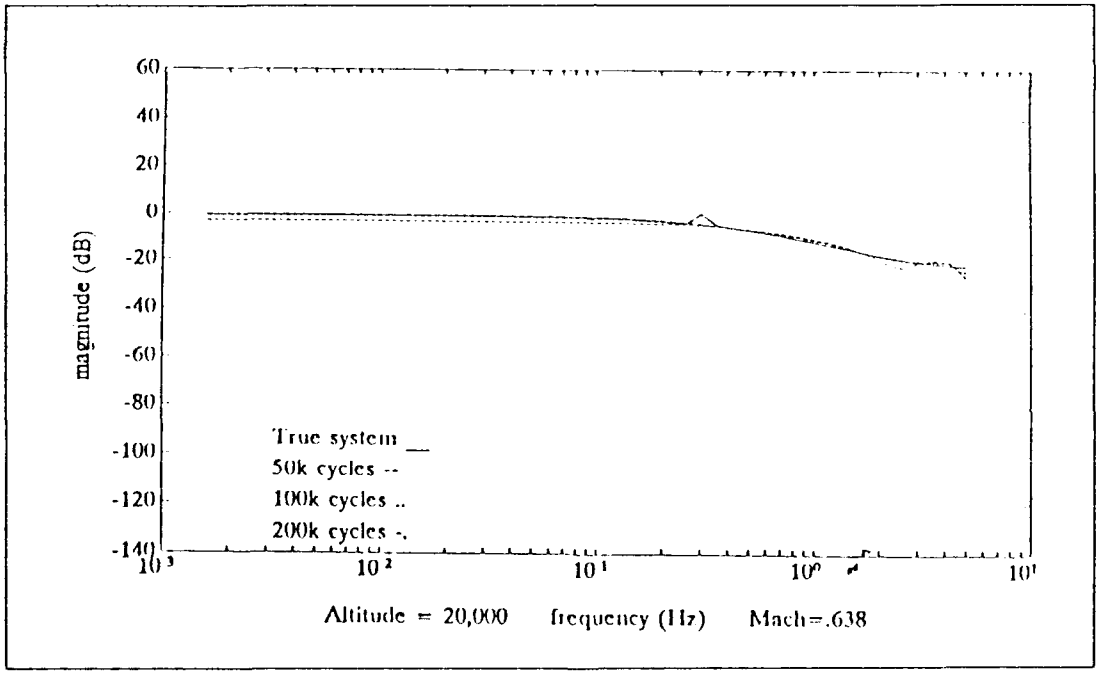

Figure 14: System and Network Frequency Response for p(t)

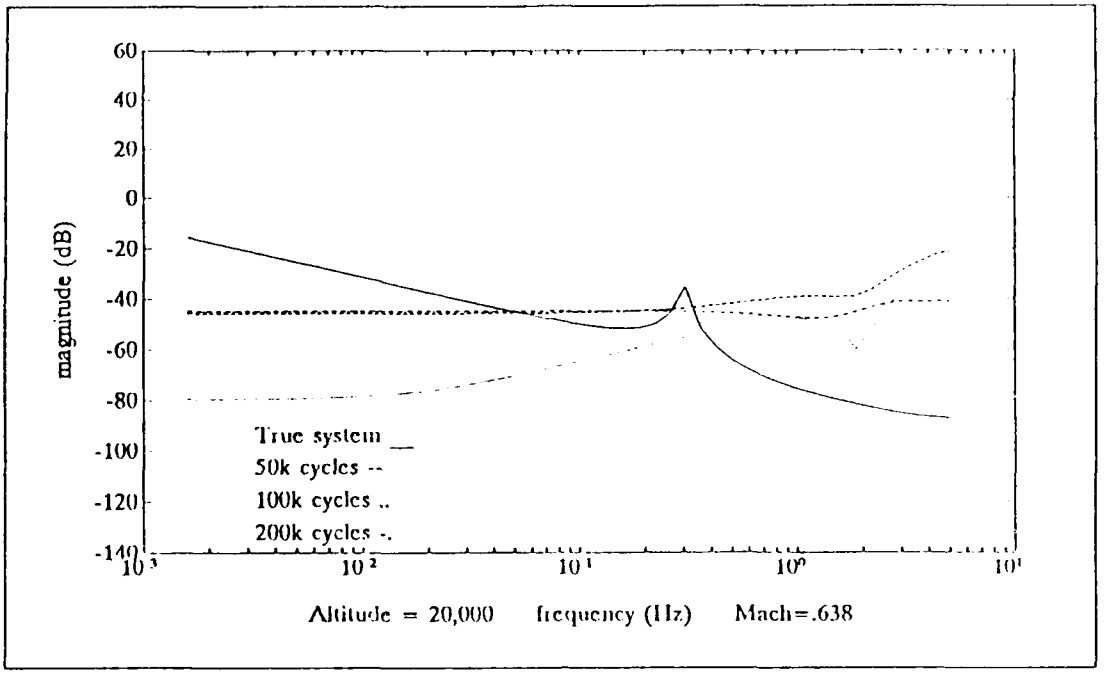

Figure 15: System and Network Frequency Response for r(t)

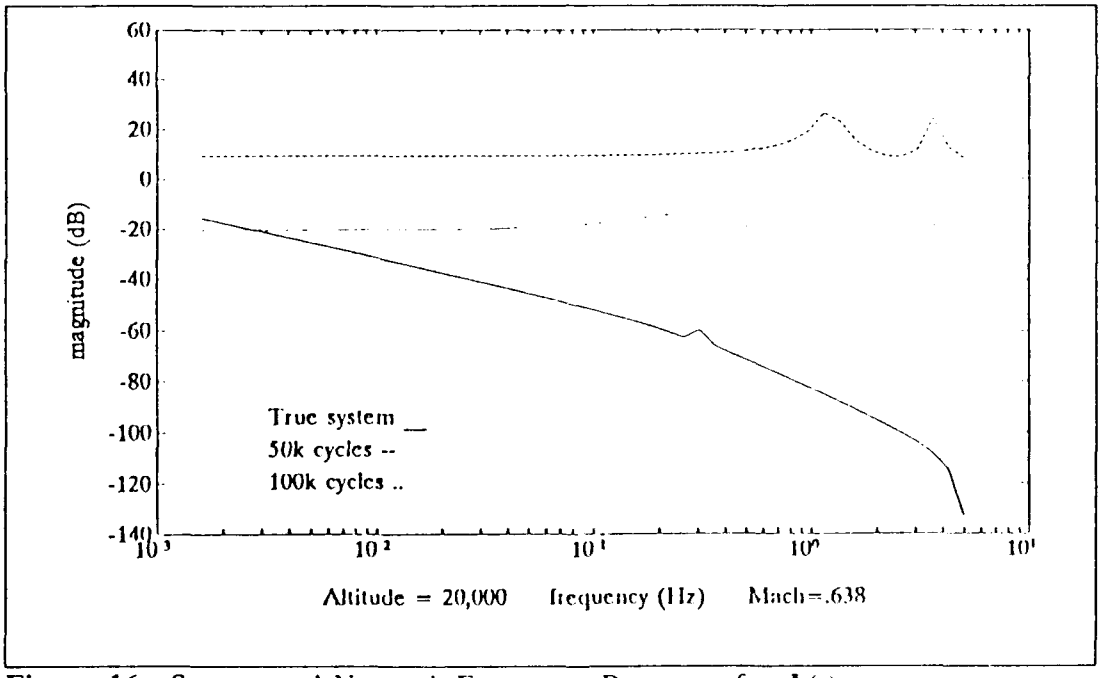

Figure 16: System and Network Frequency Response for $\Phi(t)$

from the aileron. The lack of excitation of the directional variables $\beta(t)$ and r(t) is also shown in Figure 15 where the network did a poor job of modelling the frequency response of r(t). In Figure 16, the frequency response of $\Phi(t)$ is modeled, however the proper mode shape is not obtained even through increased training.

2. Linear Network Estimation of one Parameter

Because there were large differences in the orders of magnitude between the b coefficients, and NeuralWorks Professional II only provides precision to $(10)^{-6}$, it was difficult for the network to model the entire system. Therefore a second experiment using the linear network involved a modification in which the $\beta(t)$, r(t) and $\Phi(t)$ processing elements in the network were disabled. It was thought that the increased excitation of a single parameter due to the lack of problems caused by large differences in orders of magnitude between parameters would give a better model of the p(t) parameter. Figure 17 shows that a near exact model of the frequency response and mode shape of p(t) was obtained by 50,000 cycles (5,000 seconds).

3. Network Estimation of Directional Motion

The ability of the linear neural network in estimation of the directional motion of the A-4 aircraft was tested using the same network discussed in the previous section. However, the input presented to the model by the USERIO program now simulated the directional motion of the A-4 aircraft. The system and network model frequency responses due to a rudder input are shown in Figure 18 through Figure 21. The error for each output resulting from testing this network with a pseudo random binary input was

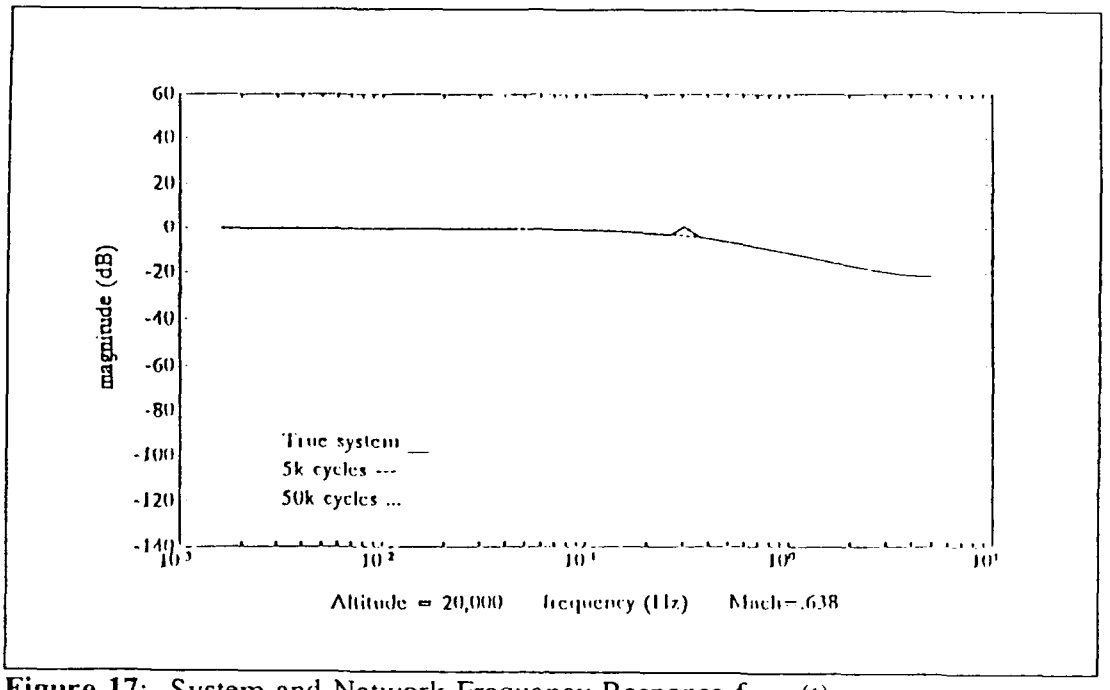

Figure 17: System and Network Frequency Response for p(t)

less than five percent for the $\beta(t)$ and p(t) variables, zero percent for r(t) and approximately forty percent for $\Phi(t)$. This would indicate that the response for all variables except $\Phi(t)$ have been learned by the network and should be modelled correctly. However, as shown in Figure 18 through Figure 20 this is not the case. Once again the networks failure to model the entire system response is due to the failure of the Neuralworks Professional II software to handle large differences in the orders of magnitude between the b coefficients in Equation (4.6). Figure 18 shows that higher frequency noise dynamics adversely effected modeling of the response above one Hertz but, by 50,000 cycles (5,000 seconds), a good approximation of the frequency response and mode shape was achieved for the $\beta(t)$ parameter. However, the model does not change much with further training and the higher frequency mode shape is not modelled

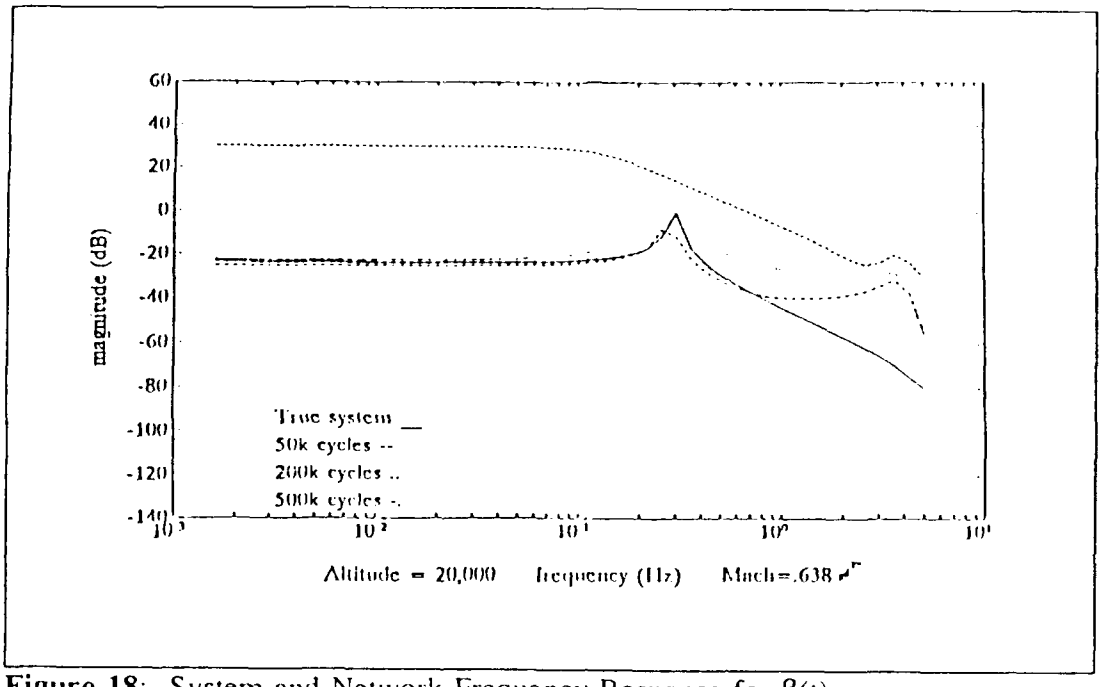

Figure 18: System and Network Frequency Response for $\beta(t)$

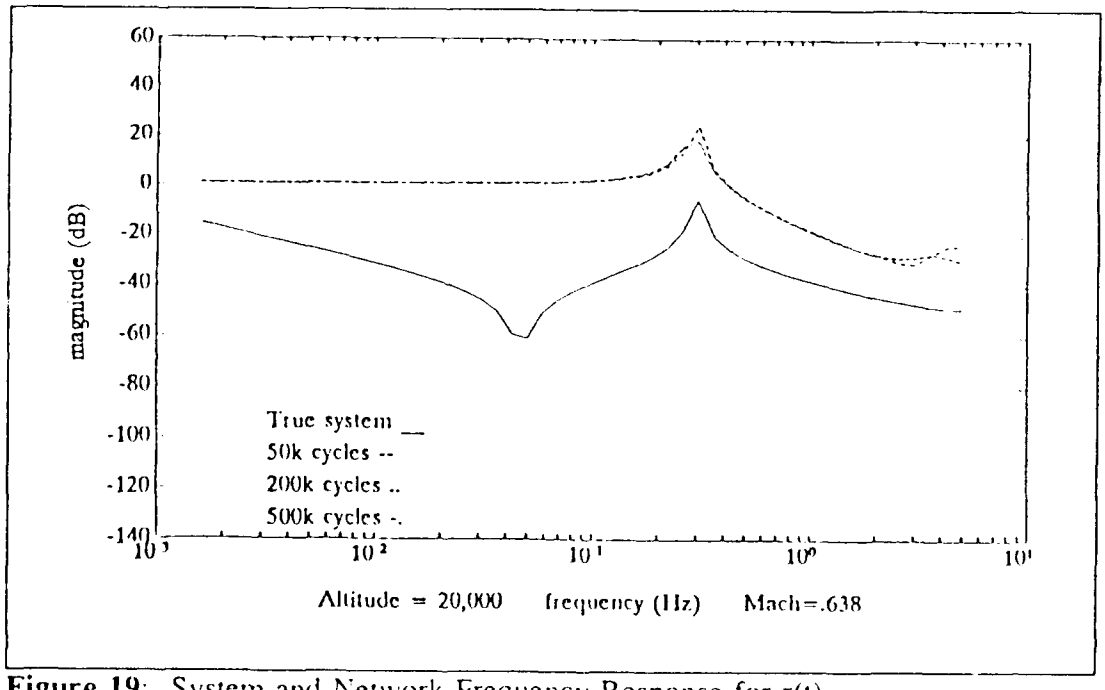

Figure 19: System and Network Frequency Response for r(t)

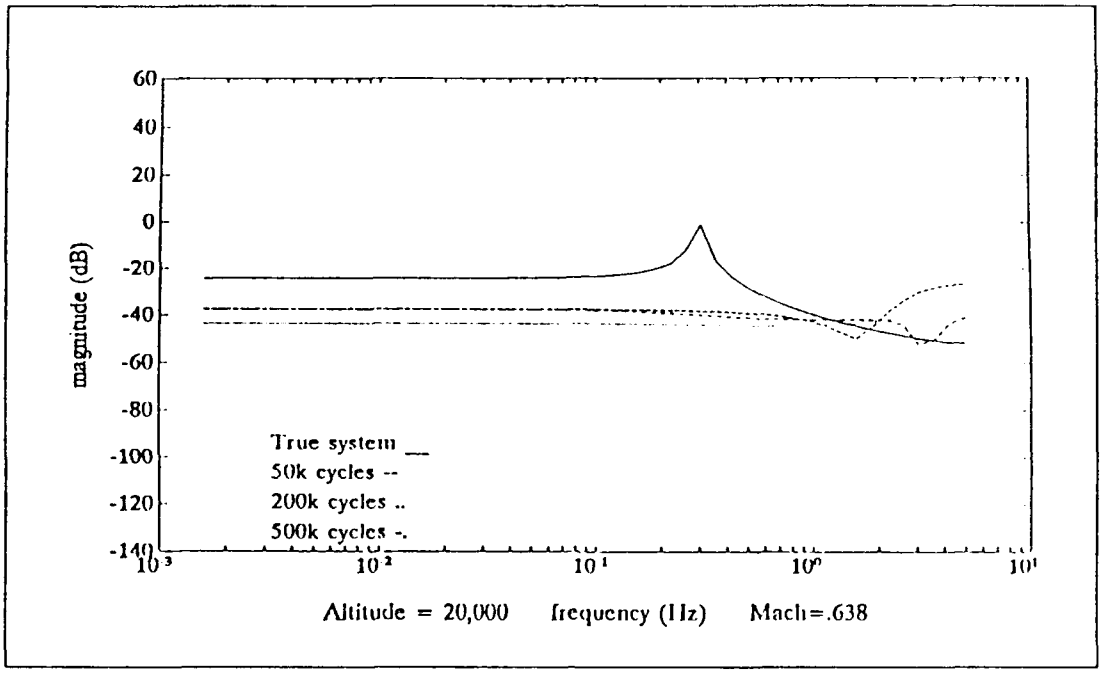

Figure 20: System and Network Frequency Response for p(t)

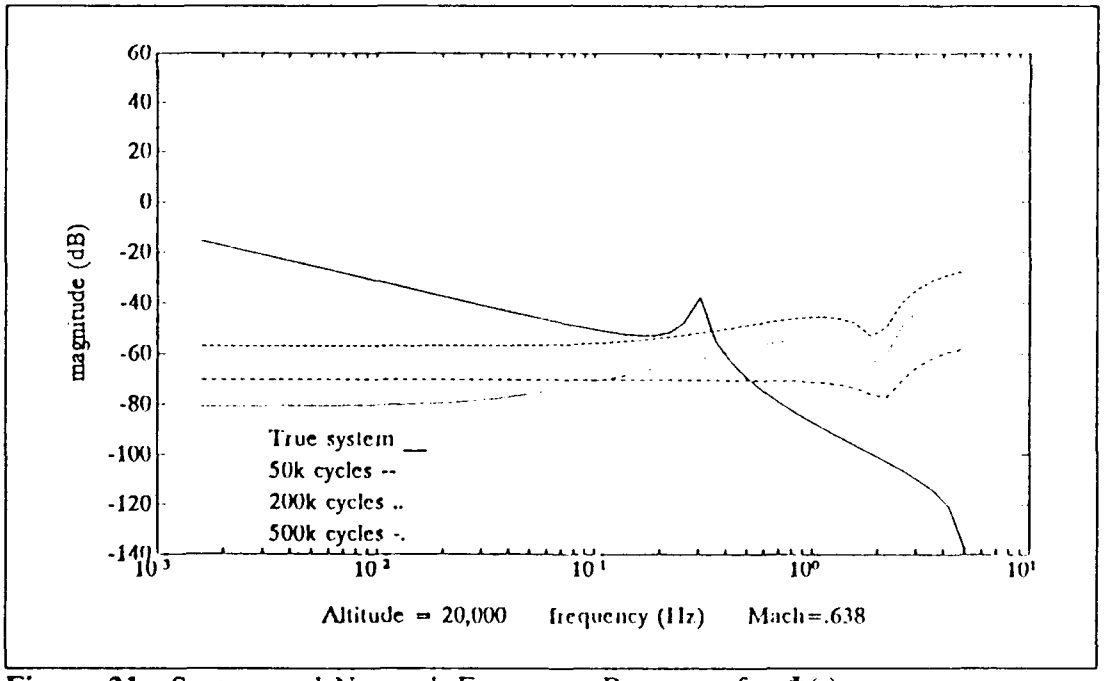

Figure 21: System and Network Frequency Response for $\Phi(t)$

even after 500,000 cycles (50,000 seconds). In Figure 19 it can be seen that the modeling of the higher frequency mode shape for the dutch roll response of r(t) improved with increased training. Although the entire mode shape, with the exception of the low frequency zero response, was modelled well, the magnitude of the networks response did not give an accurate representation of the true system response. The low frequency zero response was not modeled because the system is modelling the high frequency dyanamics of the system and does not have enough degrees of freedom to model the entire system. As expected, Figure 20 and Figure 21 show that the p(t) and $\Phi(t)$ parameters are not modeled very well during estimation of the directional motion of the A-4 aircraft. In figure 20 the low frequency mode shape is modelled, however the magnitude is less than that of the true system. The dutch roll response is not modelled and the presense of unmodelled noise dynamics in the frequencies above one Hertz is signifigant. In figure 21 it can be seen that neither the frequency response or the mode shape are modelled over the entire frequency range. An examination of the networks output for the $\Phi(t)$ parameter shows that the output is not indicative of the systems response to a random binary input. Rather it seems to be a numerical oscillation about some mean value which is unaffected by the input to the system.

4. Fully Connected Linear Neural Network

The second linear network to be examined was the network shown in Figure 11 and discussed in the previous chapter. The network was fully connected which means that all the elements in the feedback layer were connected to all the elements in the output layer. The network was trained on inputs simulating the lateral motion of the A-4

aircraft at Mach .638 and 20,000 feet. The network seems to have developed a very good model of the input-output relationship of the true system. The dynamic estimation error of all paramaters with the exception of $\Phi(t)$ went to zero in under 5,000 cycles (500 seconds). This is due to the highly overparamaterized nature of this network which allows a better ballanced representation of the system. Also, because it is fully connected, crosstalk between elements allows each output to develop dependencies on past values of other outputs.

VI. CONCLUSIONS

In this thesis a description of neural network processing and adaptive control theory has been presented. The relationship between neural networks and adaptive control theory was discussed as was the use of these two tools to develop a neural network model for adaptive control. That model was used to show that neural networks can be successfully used in the estimation of linear approximations of the lateral and directional motion of the A-4 aircraft.

The use of the neural network adaptive control structure was first demonstrated on the system of lateral motion of the A-4 aircraft. Estimation capabilities were shown using a linear neural network paramaterized as four separate transfer functions. The weights of the network were compared to the b_i and a_i terms of Equation (4.6) to show that the network built a better representation of those terms affected more by an aileron input (ie., p(t) and Φ(t)). The weights of the network were then used to generate discrete-time Bode frequency response plots for comparison with the frequency response of the true system. The network performed well at modelling the p(t) and Φ(t) variables and not the β(t) and r(t) variables. This was due to the fact that the level of excitation of the p(t) and Φ(t) variables was much larger because the only input being simulated was that from an aileron. A second experiment was conducted with this network to illustrate the problems the network has with modelling the entire system response because of large differences in the orders of magnitude between the b coefficients of Equation (4.6). The β(t), r(t),

and $\Phi(t)$ parameters were disabled and the network was run as before. The increased excitation enabled an almost exact model of p(t) to be obtained by 50,000 cycles (5,000 seconds).

A second demonstration of this structure on the system of directional motion of the A-4 aircraft reinforced the importance of excitation levels on parameter estimation. Those variables describing the directional motion due to a rudder input (ie., $\beta(t)$), and r(t), were modelled reasonably well where as the p(t) and $\Phi(t)$ parameters were not. The networks failure to model both the high and low frequency dynamics of the r(t) variable showed that the network did not have enough degrees of freedom to model the entire system.

A final fully connected version of the linear network proved to be much faster and more efficient in arriving at a representation of the true system. This was due to the highly overparamaterized nature of the network which allowed for a better ballanced representation of the system. In addition, crosstalk between elements allowed outputs to develop dependencies on past values of other outputs.

The potential for the use of neural networks in adaptive control shows much promise. Further study is needed to develop further nonlinear and linear control and estimation neural network structures. Ways of increasing the stability of the networks other than random adjustments of the learning rate should be pursued. With the continued advancements being made in modern aircraft systems, the need for better automatic control systems is apparrent. Neural networks offer one way of making those improvements.

LIST OF REFERENCES

1. Ogata, K., *Modern Control Engineering*,2d ed., Prentice-Hall, Inc., 1990.

2. Scott, R.W., *Application of Neural Networks to Adaptive Control*, Aeronautical Engineer Thesis, Naval Postgraduate School, Monterey Ca., Dec., 1989.

3. Neuralware, Inc., *Neuralworks Professional II on Sun Workstations*, 1989.

4. Rumelhart, D.E., and J. McClelland, *Parallel Distributed Processing, Explorations in the Microstructure of Cognition*, Volume 1, The MIT Press, 1987.

5. Goodwin, G.C., and K. S. Sin, *Adaptive Filtering, Prediction, and Control*, Prentice-Hall, Inc., 1984.

6. Moler, C.J., Little, and S. Bayert, *Pro-MATLAB User's Manual*, The Mathworks, Inc., 1987.

7. Nelson, R. C., *Flight Stability and Automatic Control*, McGraw - Hill, Inc., 1989.

8. McRuer, D., I. Ashdenas, and D. Graham, *Aircraft Dynamics and Automatic Control*, Princeton University Press, 1973.

APPENDIX A: NEURALWORKS PROFESSIONAL II ASSOCIATED PROGRAMS

```
/*    This file contains transfer function data for aileron inputs only     */

/*****************************************************************************
 * Source:       transfer.txt                                                *
 * Executable:   simo                                                        *
 * Version:      1.5                                                         *
 * Date:         22 November 1989                                            *
 * Author:       R. W. Scott                                                 *
 * Project:      Neural Networks in Adaptive Control                         *
 * Environment:  UNIX/SunOS C                                                *
 * Path:         eileen:/home/rscott/nworks/textfiles                        *
 * Description:  This is the header file used to define the variables        *
 *               used in the USERIO subprogram simo.  This allows easy       *
 *               reconfiguration of the executables by simply changing       *
 *               information in the header file.  Inputs include altitudes,  *
 *               airspeeds, the sampling time, selected frequencies and      *
 *               and weightings for a sum of sine waves input, labels for the*
 *               inputs, conditions, and states, and the coefficients for the*
 *               numerators and denominators of the system and various filters*
 *               used to generate filtered noise.                            *
 * Revisions:    --Inclusion of multiple input types                         *
 *               --Running the sim at twice the speed of the network         *
 *****************************************************************************/

/* Altitudes in thousands of feet */

static double altitude[5]=
        {
        0.0,0.20,0.350,0.350,0.0
        };

/* Mach Numbers */

static double mach[5]=
        {
        0.4,0.638,0.5,0.7,0.85
        };

/* Sampling Time */

static double ts={0.1};

/* Frequencies for sum of sine waves input */

static double freq[8]={
                0.005,0.09,0.11,0.65,1.5,2.75,3.0,10.0
                };

/* Frequency weighting for sum of sine waves input */

static double weight[8]={
                2.0,3.0,2.0,3.0,2.0,3.0,2.0,0.5
                };
```

```
/*  Input, condition, state, and filter labels */

static char *input_name[]={"Illegal Input","Random Binary",
        "Filtered RB","Composite Sine","Swept Square Wave-Test Only",
        "Composite Sim","Composite Time"};

static char *condition_name[]={"Illegal Condition",
        "M 0.4/SL", "M 0.638/20K", "M 0.5/35K", "M 0.7/35", "M 0.85/SL"};

static char *state_name[]={"Illegal State","Beta(t)","p(t)",
        "r(t)", "phi(t)"};

static char *filter_name[]={"Illegal Filter","0.5 Hz co",
        "0.2 Hz co","p App M 0.638/20K"};


/* Numerator coefficients */
/* Order is b1-b4,p1-p4,r1-r4,phi1-phi4 for the inner indices and
   Condition 1-Condition 5 for the outer index */
static double num[5][4][4]=
        { -9.3926871917560l1e-05,
     1.1269207025499 38e-04,
     8.8389791112142 29e-05,
    -8.6427651994736 01e-05,
     1.5617220846647 05e-01,
    -4.3291358584685 82e-01,
     4.1969082925479 47e-01,
    -1.4294945187440 69e-01,
     1.5112603562927 87e-04,
    -2.9779040530364 40e-04,
     1.7709082768080 36e-04,
    -9.6991173342830 81e-06,
     7.4349581280053 20e-05,
    -6.1480220509224 64e-05,
    -5.6458127608571 83e-05,
     6.4316107445505 64e-05,
    -1.2688133571003 85e-04,
     2.6533994721500 42e-04,
     4.3989916350906 10e-05,
    -1.8148567878117 92e-04,
     1.5763648384371 99e-01,
    -4.6480104339569 37e-01,
     4.6207673908429 48e-01,
    -1.5491217953232 04e-01,
     7.0981483787591 06e-05,
    -2.3526296186027 20e-04,
     2.5884754154770 21e-04,
    -9.3603018969434 79e-05,
     1.4495986451201 76e-05,
    -1.4546460411324 35e-05
    -1.2448603428971 69e-05,
     1.3462122399166 00e-05,
     6.7298440313958 71e-03,
     5.1976811928469 15e-03,
    -7.8546725318133 78e-03,
     6.3014122092857 06e-03,
     1.0624709218964 28e-01,
    -3.1592318570526 16e-01,
     3.1870580027252 20e-01,
    -1.0902970675690 25e-01,
```

```
     -1.439912417202382e-02,
      4.165280352077083e-02,
     -4.007056866796743e-02,
      1.285064639771638e-02,
      3.238832047398610e-04,
     -3.218367332475935e-04
     -2.938511651602305e-04,
      3.255617941835265e-04,
      3.510545510985175e-04,
     -3.016594272722273e-04,
     -3.514485515312415e-04,
      3.063414052184577e-04,
      8.470325685559521e-01,
     -2.409702137837106e-02,
      2.372696417616602e-01,
     -8.100268483354545e-02,
     -1.791587131774186e-04,
      5.494087456341390e-04,
     -5.547248529342852e-04,
      1.887628013740317e-04,
      1.951497914332023e-05,
     -1.702179099893897e-05,
     -1.635670722155424e-05,
      1.815150025163970e-05,
     -7.784614644652965e-03,
      5.747800761842115e-03,
      7.031496295012651e-03,
     -4.910648731566991e-03,
      2.851117109745740e-01,
     -6.140646745749962e-01,
      5.592805587509355e-01,
     -2.303275951505137e-01,
      3.021886108613536e-03,
     -7.749379513904309e-03,
      6.626904564018155e-03,
     -1.815375879254288e-03,
      6.763446029012243e-05,
     -1.785673815124511e-05,
     -1.410949638280812e-05,
      4.836704248745161e-05,
                 };

/* Denominator coefficients */
/* Order is den1-den4 for the inner index and Condition 1-Condition 5
   for the outer index */
static double den[5][4]=
        {
         -3.617552264583115e+00,
      5.030111197631264e+00,
     -3.185133557671352e+00,
      7.725952321069285e-01,
     -3.792822130561251e+00,
      5.420696316853906e+00,
     -3.457219170675313e+00,
      8.293442091087954e-01,
     -3.844978956470955e+00,
      5.604043890520188e+00,
     -3.669675666585049e+00,
      9.106105230285739e-01,
     -3.760009611363964e+00,
```

```
      5.409395071955029e+00,
     -3.529313846151502e+00,
      8.799325695122845e-01,
     -2.914074004278131e+00,
      3.574145070480903e+00,
     -2.219383806239528e+00,
      5.593946847263199e-01,
        };

/* Coeffieients for filtered noise terms */
/* Order is num1-num3 & den1-den2 for the inner index and filter1-
   filter3 for the outer index */

static double noise_coeff[3][5]=
        {
        2.085670251279634e-02,
        4.171340502559269e-02,
        2.085670251279634e-02,
        -1.561018075800718e+00,
        6.413515380575631e-01,
        5.063654276859733e-03 ,
        1.012730855371947e-02,
        5.063654276859733e-03,
        -1.822694925196308e+00,
        8.371816512560227e-01,
        0.0,
        -3.335605497273741e-02,
        -2.498849406787340e-02,
        -1.748500141242948e+00,
        8.340433823724368e-01
        };
```

```
/*    This file contains transfer function data for rudder inputs only     */

/*****************************************************************************
 * Source:      transfer.txt                                                 *
 * Executable:  simo                                                         *
 * Version:     1.5                                                          *
 * Date:        22 November 1989                                             *
 * Author:      R. W. Scott                                                  *
 * Project:     Neural Networks in Adaptive Control                          *
 * Environment: UNIX/SunOS C                                                 *
 * Path:        eileen:/home/rscott/nworks/textfiles                         *
 * Description: This is the header file used to define the variables         *
 *              used in the USERIO subprogram simo.  This allows easy        *
 *              reconfiguration of the executables by simply changing        *
 *              information in the header file.  Inputs include altitudes,   *
 *              airspeeds, the sampling time, selected frequencies and       *
 *              and weightings for a sum of sine waves input, labels for the *
 *              inputs, conditions, and states, and the coefficients for the *
 *              numerators and denominators of the system and various filters*
 *              used to generate filtered noise.                             *
 * Revisions:   --Inclusion of multiple input types                          *
 *              --Running the sim at twice the speed of the network          *
 *****************************************************************************/

/* Altitudes in thousands of feet */

static double altitude[5]=
        {
        0.0,0.20,0.350,0.350,0.0
        };

/* Mach Numbers */

static double mach[5]=
        {
        0.4,0.638,0.5,0.7,0.85
        };

/* Sampling Time */

static double ts={0.1};

/* Frequencies for sum of sine waves input */

static double freq[8]={
                0.005,0.09,0.11,0.65,1.5,2.75,3.0,10.0
                };

/* Frequency weighting for sum of sine waves input */

static double weight[8]={
                2.0,3.0,2.0,3.0,2.0,3.0,2.0,0.5
                };

/*  Input, condition, state, and filter labels */

static char *input_name[]={"Illegal Input","Random Binary",
        "Filtered RB","Composite Sine","Swept Square Wave-Test Only",
        "Composite Sim","Composite Time"};
```

```
static char *condition_name[]={"Illegal Condition",
        "M 0.4/SL", "M 0.638/20K", "M 0.5/35K", "M 0.7/35", "M 0.85/SL");

static char *state_name[]={"Illegal State","Beta(t)","p(t)",
        "r(t)", "phi(t)");

static char *filter_name[]={"Illegal Filter","0.5 Hz co",
        "0.2 Hz co","p App M 0.638/20K");


/* Numerator coefficients */
/* Order is b1-b4,p1-p4,r1-r4,phi1-phi4 for the inner indices and
   Condition 1-Condition 5 for the outer index */
static double num[5][4][4]=
        { 1.800083273932040e-02,
     -1.906984244706855e-02,
     -1.083224164791473e-02,
      1.189637920292630e-02,
      5.005974533658364e-02,
     -1.557746320031326e-01,
      1.564735697592665e-01,
     -5.075868309271747e-02,
     -2.949068758631812e-02,
      8.355569801024743e-02,
     -7.874398002795457e-02,
      2.465824221716073e-02,
      1.133701723912139e-04,
     -1.348051557705787e-04,
     -1.040764174029540e-04,
      1.047840637068420e-04,
      1.282768174976745e-03,
     -1.453312059030765e-04,
     -5.907510628433421e-04,
      7.611636927604692e-04,
      4.570720462542699e-03,
     -1.422345894434773e-02,
      1.438525470033047e-02,
     -4.732516218525773e-03,
     -6.894195094126143e-03,
      1.955169874457852e-02,
      5.769610846872886e-03,
     -1.842807755316045e-02,
      6.598430260407184e-06,
     -7.778000894020920e-06,
     -6.122857291845918e-06,
      6.339383324838188e-06,
      8.650627034337610e-03,
     -9.732543484972211e-03,
     -5.476291419152179e-03,
      6.551765806401710e-03,
      3.523155350109652e-02,
      1.129162889664674e-01,
     -3.802821756769237e-02,
     -6.803143835536174e-02,
      1.996683992571242e-01,
     -1.952861276856299e-01,
      6.363337184722417e-02,
      3.757891436197980e-04,
     -4.356397715552518e-04,
     -3.643866688318731e-04,
```

```
       3.904802108620764e-04,
       1.308914525035565e-02,
      -1.424512141158729e-02,
      -8.862406600178563e-03,
       1.001630183831681e-02,
       3.722835170386629e-02,
      -1.157494193649331e-01,
       1.163752393857700e-01,
      -3.785417172470429e-02,
      -9.743756038530726e-03,
       2.842606123231395e-02,
      -2.763269535968638e-02,
       8.946102177077919e-03,
       2.422209673103026e-05,
      -2.787952587457454e-05,
      -2.407882636212832e-05,
       2.344827517997139e-05,
       8.954306417501723e-02,
      -7.686065616734794e-02,
      -6.310568929134819e-02,
       5.042267913457954e-02,
       1.381716402232911e-01,
      -4.428201053637051e-01,
       4.432764525603321e-01,
      -1.386279874199177e-01,
      -1.150444201425271e-01,
       3.047982225836359e-01,
      -2.659113340054455e-01,
       7.607349598960700e-02,
       2.505311791232145e-04,
      -2.080249895417552e-04,
       2.038679740665739e-04,
                 };

/* Denominator coefficients */
/* Order is den1-den4 for the inner index and Condition 1-Condition 5
   for the outer index */
static double den[5][4]=
        {
         -3.617552264583112e+00,
       5.030111197631257e+00,
      -3.185133557671345e+00,
       7.725952321069264e-01,
      -3.792822130561248e+00,
       5.420696316853899e+00,
      -3.457219170675306e+00,
       8.293442091087930e-01,
      -3.899605053343789e+00,
       5.768525031214528e+00,
      -3.833555439643835e+00,
       9.646017086430867e-01,
      -3.760009611363958e+00,
       5.409395071955011e+00,
      -3.529313846151485e+00,
       8.799325695122787e-01,
      -2.914074004278134e+00,
       3.574145070480905e+00,
      -2.219383806239528e+00,
       5.593946847263198e-01
        };
```

APPENDIX B: MATLAB M-FILE

```
% CONTINUOUS STATE SPACE TO DISCRETE TRANSFER FUNCTION
CONVERSION
%
% Altitude = sea level
% Mach# = .4
% u = 446.6 fps
%
format short e
% Initial plant matrix
a=[-.243 0. -1. .072; -27.3 -1.699 .948 0.;
14.9 .065 -.638 0.; 0. 1. 0. 0.]
%
% Initial control matrix
b=[0.; 16.526; .0671; 0]
%
% Convert b from radians to degrees
b=b*pi/180;
% Compute scaling matrix 'c'
c=eye(a); d=[0 0 0 0]';
w=logspace(-3,2);
[mag,phase]=bode(a,b,c,d,1,w);
c=max(mag);
x=[1,1,1,1];
c=(x./c)
c=diag(c);
%
%Balance a,b, and c matrices
[ab,bb,cb]=obalreal(a,b,c);
%
% Convert to discrete time
t=.1;
[ad,bd]=c2d(ab,bb,t);
%
% Convert to transfer function
[NUM,DEN]=ss2tf(ad,bd,cb,d,1)
```

APPENDIX C: CONTINUOUS STATE SPACE EQUATIONS AND DISCRETE MATRIX POLYNOMIALS FOR AN AILERON INPUT

Sampling Time of 0.1 Seconds

FLIGHT CONDITION 1

Altitude = sea level

Mach# = .4

u = 446.6 fps

a =

-2.4300e-001	0	-1.0000e+000	7.2000e-002
-2.7300e+001	-1.6990e+000	9.4800e-001	0
1.4900e+001	6.5000e-002	-6.3800e-001	0
0	1.0000e+000	0	0

b =

0
1.6526e+001
6.7100e-002
0

NUM =

0	-9.3927e-005	1.1269e-004	8.8390e-005	-8.6428e-005
0	1.5617e-001	-4.3291e-001	4.1969e-001	-1.4295e-001
0	1.5113e-004	-2.9779e-004	1.7709e-004	-9.6991e-006
0	7.4350e-005	-6.1480e-005	-5.6458e-005	6.4316e-005

DEN =

1.0000e+000	-3.6176e+000	5.0301e+000	-3.1851e+000	7.7260e-001

FLIGHT CONDITION 2

Altitude = 20,000 ft.

Mach# = .638

u = 660 fps

a =

-8.2900e-002	0	-1.0000e+000	4.8800e-002
-4.5460e+000	-1.6990e+000	1.7170e-001	0
3.3820e+000	-6.5400e-002	-8.9300e-002	0
0	1.0000e+000	0	0

b =

0
2.7276e+001
3.9520e-001
0

NUM =

0	-1.2688e-004	2.6534e-004	4.3990e-005	-1.8149e-004
0	1.5764e-001	-4.6480e-001	4.6208e-001	-1.5491e-001
0	7.0981e-005	-2.3526e-004	2.5885e-004	-9.3603e-005
0	1.4496e-005	-1.4546e-005	-1.2449e-005	1.3462e-005

DEN =

1.0000e+000	-3.7928e+000	5.4207e+000	-3.4572e+000	8.2934e-001

FLIGHT CONDITION 3

Altitude = 35,000 ft.

Mach# = .5

u = 487 fps

a =

-8.6400e-002	0	-1.0000e+000	6.6000e-002
-9.0300e+000	-5.6200e-001	4.0400e-001	0
6.5200e+000	-6.7600e-002	2.8800e-001	0
0	1.0000e+000	0	0

b =

-9.0000e-004
5.4000e+000
-7.5900e-001
0

NUM =

0	6.7298e-003	-5.1977e-003	-7.8547e-003	6.3014e-003
0	1.0625e-001	-3.1592e-001	3.1871e-001	-1.0903e-001
0	-1.4399e-002	4.1653e-002	-4.0071e-002	1.2851e-002
0	3.2388e-004	-3.2184e-004	-2.9385e-004	3.2556e-004

DEN =

1.0000e+000	-3.8996e+000	5.7685e+000	-3.8336e+000	9.6460e-001

FLIGHT CONDITION 4

Altitude = 35,000 ft.

Mach# = .7

u = 681 fps

a =

-1.2210e-001	0	-1.0000e+000	4.7000e-002
-2.2700e+001	-8.1900e-001	5.6000e-001	0
1.1990e+001	3.3400e-002	-3.3800e-001	0
0	1.0000e+000	0	0

b =

0
1.1770e+001
-2.7500e-001
0

NUM =

0	3.5105e-004	-3.0166e-004	-3.5145e-004	3.0634e-004
0	8.4703e-002	-2.4097e-001	2.3727e-001	-8.1003e-002
0	-1.7916e-004	5.4941e-004	-5.5472e-004	1.8876e-004
0	1.9515e-005	-1.7022e-005	-1.6357e-005	1.8152e-005

DEN =

1.0000e+000	-3.7600e+000	5.4094e+000	-3.5293e+000	8.7993e-001

FLIGHT CONDITION 5

Altitude = sea level

Mach# = .85

u = 950fps

a =

```
 -5.7500e-001            0 -1.0000e+000  3.4000e-002
 -1.1760e+002 -3.8300e+000  1.9300e+000            0
  6.8000e+001  4.5600e-002 -1.4040e+000            0
            0  1.0000e+000            0            0
```

b =

```
            0
  6.4400e+001
  5.0500e+000
            0
```

NUM =

```
 0 -7.7846e-003  5.7478e-003  7.0315e-003 -4.9106e-003
 0  2.8511e-001 -6.1406e-001  5.5928e-001 -2.3033e-001
 0  3.0219e-003 -7.7494e-003  6.6269e-003 -1.8154e-003
 0  6.7634e-005 -1.7857e-005 -1.4109e-005  4.8367e-005
```

DEN =

```
1.0000e+000 -2.9141e+000  3.5741e+000 -2.2194e+000 5.5939e-001
```

APPENDIX D: CONTINUOUS STATE SPACE EQUATIONS AND DISCRETE MATRIX POLYNOMIALS FOR A RUDDER INPUT

Sampling time of 0.1 seconds

FLIGHT CONDITION 1

Altitude = sea level

Mach# = .4

u = 446.6 fps

a =

-2.4300e-001	0	-1.0000e+000	7.2000e-002
-2.7300e+001	-1.6990e+000	9.4800e-001	0
1.4900e+001	6.5000e-002	-6.3800e-001	0
0	1.0000e+000	0	0

b =

6.1785e-004
1.2846e-001
-1.1414e-001
0

NUM =

0	1.8001e-002	-1.9070e-002	-1.0832e-002	1.1896e-002
0	5.0060e-002	-1.5577e-001	1.5647e-001	-5.0759e-002
0	-2.9491e-002	8.3556e-002	-7.8744e-002	2.4658e-002
0	1.1337e-004	-1.3481e-004	-1.0408e-004	1.0478e-004

DEN =

1.0000e+000 -3.6176e+000 5.0301e+000 -3.1851e+000 7.7260e-001

FLIGHT CONDITION 2

Altitude = 20,000 ft.

Mach# = .638

u = 660 fps

a =

```
 -8.2900e-002            0 -1.0000e+000  4.8800e-002
 -4.5460e+000 -1.6990e+000  1.7170e-001            0
  3.3820e+000 -6.5400e-002 -8.9300e-002            0
            0  1.0000e+000            0            0
```

b =

```
  2.0246e-004
  1.0050e-002
 -2.3771e-002
            0
```

NUM =

```
 0  1.2828e-003 -1.4533e-003 -5.9075e-004  7.6116e-004
 0  4.5707e-003 -1.4223e-002  1.4385e-002 -4.7325e-003
 0 -6.8942e-003  1.9552e-002 -1.8428e-002  5.7696e-003
 0  6.5984e-006 -7.7780e-006 -6.1229e-006  6.3394e-006
```

DEN =

```
1.0000e+000 -3.7928e+000  5.4207e+000 -3.4572e+000 8.2934e-001
```

FLIGHT CONDITION 3

Altitude = 35,000 ft.

Mach# = .5

u = 487 fps

a =

```
 -8.6400e-002            0 -1.0000e+000  6.6000e-002
 -9.0300e+000 -5.6200e-001  4.0400e-001            0
  6.5200e+000 -6.7600e-002  2.8800e-001            0
            0  1.0000e+000            0            0
```

b =

```
  2.6128e-004
  3.9270e-002
 -4.7298e-002
            0
```

NUM =

```
 0  8.6506e-003 -9.7325e-003 -5.4763e-003  6.5518e-003
 0  3.5232e-002 -1.1012e-001  1.1292e-001 -3.8028e-002
 0 -6.8031e-002  1.9967e-001 -1.9529e-001  6.3633e-002
 0  3.7579e-004 -4.3564e-004 -3.6439e-004  3.9048e-004
```

DEN =

```
1.0000e+000 -3.8996e+000  5.7685e+000 -3.8336e+000 9.6460e-001
```

FLIGHT CONDITION 4

Altitude = 35,000 ft.

Mach# = .7

u = 681 fps

a =

-1.2210e-001	0	-1.0000e+000	4.7000e-002
-2.2700e+001	-8.1900e-001	5.6000e-001	0
1.1990e+001	3.3400e-002	-3.3800e-001	0
0	1.0000e+000	0	0

b =

3.6128e-004
9.1979e-002
-8.4648e-002
0

NUM =

0	1.3089e-002	-1.4245e-002	-8.8624e-003	1.0016e-002
0	3.7228e-002	-1.1575e-001	1.1638e-001	-3.7854e-002
0	-9.7438e-003	2.8426e-002	-2.7633e-002	8.9461e-003
0	2.4222e-005	-2.7880e-005	-2.4079e-005	2.3448e-005

DEN =

1.0000e+000	-3.7600e+000	5.4094e+000	-3.5293e+000	8.7993e-001

FLIGHT CONDITION 5

Altitude = sea level

Mach# = .85

u = 950 fps

a =

-5.7500e-001	0	-1.0000e+000	3.4000e-002
-1.1760e+002	-3.8300e+000	1.9300e+000	0
6.8000e+001	4.5600e-002	-1.4040e+000	0
0	1.0000e+000	0	0

b =

1.5673e-003
6.4577e-001
-4.7298e-001
0

NUM =

0	8.9543e-002	-7.6861e-002	-6.3106e-002	5.0423e-002
0	1.3817e-001	-4.4282e-001	4.4328e-001	-1.3863e-001
0	-1.1504e-001	3.0480e-001	-2.6591e-001	7.6073e-002
0	2.5053e-004	-3.3041e-004	-2.0802e-004	2.0387e-004

DEN =

1.0000e+000 -2.9141e+000 3.5741e+000 -2.2194e+000 5.5939e-001

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Code AA Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5002	1
4.	Professor D.J. Collins Code AACo Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5002	5
5.	Professor L.V. Schmidt Code AASc Department of Aeronautics and Astronautics Naval Postgraduate School Monterey, California 93943-5002	1
6.	Mr. Charles Cumbow Code FW82 Force Warfare Aircraft Test Directorate Naval Air Test Center Patuxent River, MD 20670	1
7.	Mr. Donald Nedresky Code FW82S Force Warfare Aircraft Test Directorate Naval Air Test Center Patuxent River, MD 20670	2